FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

DESENVOLVIMENTO DE UMA PLATAFORMA ELABORADA PARA PROJETOS DE SISTEMAS EMBARCADOS RECONFIGURÁVEIS (*ARM7* E *FPGA*)

SAMUEL CÉSAR DA CRUZ JÚNIOR

ORIENTADOR: CARLOS HUMBERTO LLANOS QUINTERO

DISSERTAÇÃO DE MESTRADO EM SISTEMAS MECATRÔNICOS

PUBLICAÇÃO:
BRASÍLIA, AGOSTO DE 2012

FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

DESENVOLVIMENTO DE UMA PLATAFORMA ELABORADA PARA PROJETOS DE SISTEMAS EMBARCADOS RECONFIGURÁVEIS (*ARM7* E *FPGA*)

SAMUEL CÉSAR DA CRUZ JÚNIOR

DISSERTAÇÃO A SER SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA MECÂNICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISÍTOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM SISTEMAS MECATRÔNICOS.

APROVADA POR:

____

Prof. Dr. Carlos Humberto Llanos Quintero (ENM-UnB)
(Orientador)

____

Prof. Ricardo Pezzuol Jacobi (CIC-UnB)
(Examinador Interno)

____

Prof. Pedro de Azevedo Berger (CIC-UnB)

BRSÍLIA, AGOSTO DE 2012

FICHA CATALOGRÁFICA

CRUZ JUNIOR, SAMUEL C.	
Desenvolvimento de uma plataforma elaborada para projetos de sistemas embarcados reconfiguráveis (*ARM7* e *FPGA*) [Distrito Federal] 2012.	
xxi, 1p., 210 x 297 mm (ENM/FT/UnB, Mestre, sistemas Mecatrônicos, 2012).	
Dissertação de Mestrado – Universidade de Brasília. Faculdade de Tecnologia.	
Departamento de Engenharia Mecânica.	
1. Sistemas Embarcados	2. Computação Ubíqua
3. Sistemas Reconfiguráveis	4. ARM7 & FPGA
I. ENM/FT/UnB	II. Titulo(série)

REFERÊNCIA BIBLIOGRÁFICA

Cruz Jr, S. C.. (2012). Desenvolvimento de uma plataforma elaborada para projetos de sistemas embarcados reconfiguráveis (*ARM7* e *FPGA*). Dissertação de Mestrado em Sistemas Mecatrônicos, Publicação ENM.DM-52/12, Departamento de Engenharia Mecânica, Universidade de Brasília, Brasília, DF, p. 1.

CESSÃO DE DIREITOS

Samuel César da Cruz Júnior

sccjunior@gmail.com

Brasília – DF – Brasil.

As pessoas não sabem o que querem até você mostrar a elas.

Steven Paul Jobs

AGRADESCIMENTOS

Acima de tudo à Deus que me conduz e sustenta desde o ventre de minha mãe. Dá-me força e o ânimo em momentos de dificuldade e cuida dos menores detalhes em minha existência.

Agradeço, de forma muito especial ao Prof. Dr. Carlos Humberto Llanos Quintero que, de maneira muito sábia, vem sendo meu mentor e minha referência em termos de conhecimento, ética e seriedade dentro da Universidade de Brasília durante anos. Agradeço de coração pela confiança, generosidade e, sobretudo, pela amizade.

À minha esposa, Ketlen César, e filha, Emily César, que, com muito amor e cuidado, sempre souberam entender e me apoiar nas maiores dificuldades e responsabilidades advindos deste compromisso. Pelas indispensáveis e valiosas contribuições feitas por minha esposa ao longo deste trabalho.

Aos meu pais, Samoel e Cleoni César, que nunca hesitaram em oferecer-me o melhor de seus recursos financeiros, éticos, morais e cristãos. Por me conduzirem pelos caminhos da verdade e da vida. Aos meus sobrinhos Estevão e Jordana pelo comportamento exemplar durante a fase de elaboração deste relatório. À minha irmã, Denise César, e meu cunhado, Enoque Castro, pela amizade e companheirismo.

À minha família e amigos por tudo que ensinaram e por todas as contribuições, direta ou indiretamente dadas. Em especial, ao MsC Magno Batista Corrêa que de maneira muito espirituosa dedicou noites e fins de semanas a ajudar incondicional e irrestritamente a atingir os resultados ora alcançados. Da mesma forma, sua esposa e filha pela compreensão e bondade.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico pelos importantes aportes financeiros, indispensáveis para a execução desta pesquisa.

RESUMO

Nas últimas décadas tem-se observado um aumento exponencial de dispositivos eletrônicos dedicados ao conforto, comodidade, diversão ou segurança pessoal. A massificação dos equipamentos eletrônicos já abrange mercados de consumo e de capitais, como: indústria automobilística, áudio e vídeo, eletrodomésticos, bens de consumo, robótica, entre outros. A grande possibilidade de interação entre o homem e as máquinas é o combustível para o desenvolvimento da eletrônica dedicada a uma ou algumas aplicações, os quais são chamados de sistemas embarcados ou embutidos. A partir daí os sistemas ubíquos têm ganhado mercado como soluções computacionais pela interação sutil e constante entre homens e equipamentos eletrônicos de maneira muito natural no cotidiano das pessoas. Buscando atender a uma demanda por um *hardware* de aquisição, processamento e controle de sinais para ambientes não industriais foi elaborada uma plataforma de desenvolvimento. Este *hardware* possui dois núcleos de processamento, uma Maquina *RISC* Avançada 7 - *ARM7*[1] e um Arranjo de Portas Programáveis em Campo - *FPGA*[2] com interface externa com suporte a protocolos específicos (*SPI*, *RS232*, *JTAG*, *USB*, *Ethernet*) e ainda interface com o usuário por meio de botões, potenciômetro e *LEDs*. Ademais, os dois núcleos podem trabalhar em conjunto ou separadamente, conforme a necessidade do usuário. A combinação da versatilidade e baixo custo dos processadores *ARM7* (amplamente utilizados em sistemas embarcados, com a multifuncionalidade) com a flexibilidade e alta capacidade de processamento dos *FPGAs* forma uma interessante combinação para os mais diversos projetos voltados para controle e automação de sistemas.

Palavras chaves: sistemas embarcados, computação ubíqua, sistemas reconfiguráveis, ARM7 & FPGA, automação ambiental.

[1] Em inglês, Advanced RISC Machine.
[2] Em inglês, Field-Programmable Gate Array.

ABSTRACT

Over the last decades it has been observed an exponential increase of electronic devices dedicated to provide comfort, convenience, fun and safety to people. The popularity of electronic equipment comprises consumer and capital markets, such as: automobile, audio and video industries, household appliances and consumer goods, among others. The vast possibility of interaction between human and machine is the fuel for the development of electronic devices dedicated to one or more applications, which are called embedded or built-in systems. From this point on, the ubiquitous systems has gained market as computer solutions for constant and subtle interaction between humans and electronic equipment in a very natural way of an everyday life. To achieve the requirements for specific tasks, namely hardware acquisition, processing and controlling signals for non-industrial environments, a development board kit has been designed. This hardware has two processing cores, an ARM7 (Advanced RISC Machine) and a FPGA (Field-Programmable Gate Array), which have external interfaces supporting specific protocols (e.g. SPI, RS232, JTAG, USB, Ethernet), and also user interfaces through push buttons, potentiometer and LEDs. Additionally, the two devices can work together or separately, as required by the user. The combination of versatility and low cost of ARM7 processor (widely used in embedded systems with multi-functionality) and the high flexibility and processing power of FPGAs shows up an interesting solution for projects related to control and automation systems.

Key words: embedded systems, ubiquitous computing, reconfigurable systems, ARM7 & FPGA, automation environment.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIMBOLOS, SIGLAS E ABREVIATURAS

ADC: Analog Digital Converter – Conversor Analógico Digital.

AHDL: *Altera hardware descriptive language* – Linguagem de descrição de *hardware* da Altera.

ASIC: *Application Specific Integrated Circuit* – Circuitos integrados para aplicações específicas.

API: *Application soniProgram Interface* – Interface de aplicação programação*.*

ARM: *Advanced RISC Machine* – Dispositivo avançado *RISC.*

ARP: *Address Resolution Protocol* – Protocolo de resolução de endereçamento

BOD: *Brown-Out Detect* – Detector de *Brown-Out*.

BSL: *Boot Loader Select* – Seletor de *Boot Loader*.

BPI: *Byte-Wide Peripheral Interface* – Interface periférica de largura de *byte.*

CAD: Computer-aided design – Desenho assistido por computador.

CCM: *Custom Computing Machine* – Sistemas Computacionais Customizados

CLP: Controlador Lógico Programável.

CPLDs: *Complex Programmable Logic Devices* – Dispositivos lógicos de programação complexa

CPU: Central Processing Unit – Unidade central de processamento

CSL: Chip Select – Seletor de chip.

DAC : *Digital Analog Converter* – Conversor digital analógico.

D*FPGAs*: *Dinamically Field Programmable Gate Arrays* - Arranjo de portas dinamicamente programáveis em campo

DHCP: *Dynamic Host Configuration Protocol* – Protocolo de configuração dinâmica de hospedeiro.

DSP: *Disigned System Processor* – Sistema projetado de processamento.

EDA: Electronic Design Automation – Desenho eletrônico automatizado.

EEPROM: *Electrically Erasable Programmable Read-Only Memory* – Memória de somente leitura, apagável e programável eletricamente.

EIA: *Electronic Industries Association* – Associação das indústrias eletrônicas.

EPROM: *Erasable Programmable Only Memory* – Memória Programável Apagável.

FLASH: Tipo de memória EEPROM

FPGA: *Field Programmable Gate Array* - Arranjo de Portas Programável em Campo.

FCCM: *FPGA – based custom computing machine – Hardware* reconfigurável baseado em sistemas computacionais customizados.

FFT: *Fast Fourier Transform* - Transformada rápida de Fourier

GCLK: Global Clock.

GPP: General Purpose Processors – Processadores de Propósito Geral.

HDL: *Hardware Description Language* – Linguagem de descrição de *Hardware*.

I^2C: Inter Integrated Circuit – Entre Circuitos Itegrados.

I/O: Input/Output – Entrada e saída.

IAP: In-Application Programming – Programação Durante a Aplicação.

IEEE: Instituto de Engenheiros Eletricistas e Eletrônicos.

IP: Internet Protocol – Protocolo de Internet.

IP-cores: Intelectual Property Cores – Núcleos de Propriedade Intelectual.

*ISA: **I**ndustry **S**tandard **A**rchitecture* – Sistema de informações da arquitetura.

ISO: International Standart Organization – Organização de Padronização Internacional.

ISP: In-Circuit Programming – Programação no Circuito.

JTAG: The Joint Test Access Group – Grupo da Junção de Teste e Acesso.

LED: Light Emitter Diode - Diodo Emissor de Luz.

MAC: Media Access Control – Controle de Midia de Acesso.

RAM: Random Access Memory – Memória de Acesso Randômico.

ROM: *Read-Only Memory* – Memória apenas de leitura.

PB: *Push Button* – Botão de pressionamento ou táctil.

PC: *Personal Computer* – Computador pessoal.

PCB: *Printed Circuit Board* – Placa de Circuito Impresso.

PLDs: *Programmable Logic Devices* – Dispositivos de lógica programável.

PLL: Phase Locked Loop – Volta de fase travada.

POR: *Power-On Reset* – Reset de Inicialização.

PWM: *Pulse With Modulation* – Modulação por Largura de Pulso.

RISC: *Reduced Instruction Set Computer* – Computador com Instruções Reduzidas

SoC: *System-on-Chip* – Sistema em Chip.

RS: *Recommended Standard* – Padrão Recomendado.

RTC: Real-Time Clock – Relógio de Tempo Real.

SCL: Serial Clock

SDA: Serial Data

SMD: S*urface Mount Device* – Dispositivo para Montar em Superfície.

SRAM: *Static Random Access Memory* – Memória Estática de Acesso Randômico.

SPI: Serial Peripheral Interface Bus – Interface Serial Periférica.

TAP: *Test Access Port controller –* Controlador de Teste de Acesso de Porta

TIC: Tecnologias da Informação e Comunicação

TLR: Test Logic Reset

TTM: Time to Market – Tempo até o Mercado.

UART: *Universal Asynchronous Receiver/Transmitter* – Transmissor e Receptor Assíncronos Universais.

UDP: User Datagram Protocol – Protocolo de Datagrama de Usuário

USB: Universal Serial Bus

VHDL: *Very Hight Description Language* – Linguagem de Muito Elevada Descrição

CAPÍTULO 1
INTRODUÇÃO

A automação já se tornou essencial para a sociedade contemporânea. Nos últimos anos houve um desenvolvimento intenso da eletrônica, computação e sistemas mecânicos de precisão, que difundiram amplamente os dispositivos automáticos na sociedade moderna. Dando subsidio a estes dispositivos, destacam-se os circuitos integrados (CIs) que estão oferecendo cada vez melhores índices de desempenho, incluindo também maior quantidade de transistores por unidade de área. Os microprocessadores, por exemplo, que nos anos de 1980 operavam a uma velocidade na faixa de KHz, hoje, já são comercializados em GHz (Shima, 2005). Esse desenvolvimento fomentou a evolução das áreas de microeletrônica, eletrônica digital e engenharias, especialmente, de controle e automação (Srivastava *et al.*, 1998).

Sistemas de tempo real e embarcados representaram, historicamente, uma pequena parcela se comparados àqueles de propósito gerais, também conhecidos como *GPP*[3]. No entanto, o avanço da microeletrônica e dos sistemas reconfiguráveis permitiu que os sistemas embarcados fossem incorporados à vida das pessoas com significativas melhorias de funcionalidade, redução de custos e portabilidade. Essas tecnologias trabalham como facilitadoras para o desenvolvimento industrial, bem como transformadoras de organizações e mercados (Pereira; Carro, 2007).

Resultado dessa evolução é a mudança no cotidiano das pessoas, pela inserção de dispositivos eletrônicos nos mais diversos utensílios, tais como, *smartphones*, aparelhos de áudio, vídeo, máquinas de lavar, automóveis, entre outros (Buchanan, 2001).

A interação harmônica entre homem e eletrônica, assim como o avanço na elaboração de produtos baseados em *sistemas embarcados*, com características de sistemas em tempo real, resultou na elaboração de vários conceitos de aplicações, tais como domótica e computação ubíqua (Weiser, 1991). Esses sistemas também são conhecidos como computação *pervasiva*, *tecnologia calma*, *coisas que pensam* ou ainda *inteligência ambiental* (Satyanarayanan, 2001), os quais despertam grande interesse da comunidade acadêmica, e, especialmente, do Laboratório de Eletrônica do GRACO.

[3] Em ingles: *General Purpose Processors*

O processo de desenvolvimento desses dispositivos/sistemas (para aplicações em automação, domótica e computação ubíqua) depende da combinação adequada entre *software* e *hardware*. Muitas vezes os projetistas optam por utilizar *hardwares* prontos para acelerar a fase de pesquisa e desenvolvimento de novos produtos. A elaboração do *software*, torna-se mais rápida e barata, em relação ao *hardware,* quando existe uma plataforma para desenvolvimento pronta (Dagnino, 2001). Com um *hardware* dedicado a testes e desenvolvimento em mãos (para sistemas embarcados), é possível evoluir e amadurecer o produto até que, ao final, seja elaborado um *hardware ad hoc,* nada além do suficiente para a aplicação desejada.

Sistemas embarcados são ideais para ambientes que exijam aspectos como flexibilidade, bom desempenho, baixo consumo de energia e potência, e ainda portabilidade (Saint-Jean *et al.*, 2007). Sistemas embarcados geralmente trabalham com *clocks* de baixa frequência, normalmente abaixo dos 500 MHz, e por este motivo podem se beneficiar do uso de FPGAs para mapeamento de algoritmos diretamente em *hardware.* Dessa forma, a alta flexibilidade do sistema – permitindo mapear algoritmos diretamente em *hardware*, via *FPGA* – pode compensar o que seria uma relativa perda de desempenho pela baixa frequência de *clock* (Hartenstein, 2007).

Conforme estudos realizados por Hartenstein (2006) a computação reconfigurável (baseada em FPGAs) oferece uma drástica redução de consumo de energia pela mesma capacidade de processamento, se comparadas às tradicionais maquinas baseadas no paradigma de von Neumann.

A exploração das múltiplas possibilidades de desenvolvimento de sistemas envolvendo o uso, em conjunto, de processadores de uso geral (*GPP – General Purpose Processors*) e *FPGAs* vem sendo explorada por meio dos chamados *Sistemas em Chip* ou *SoC*[4](Kazmierkowski, 2011). A fronteira do conhecimento desta área vem recebendo contribuições de diferentes grupos de pesquisa pelo mundo. Basicamente, consiste na elaboração de sistemas complexos, com múltiplos processadores e módulos específicos de *hardware*, encapsulados em um único dispositivo e conectados por meio de redes *intra-chip* (Bartic *et al.*, 2003).

No caso de placas de desenvolvimento de sistemas, envolvendo *GPPs* e *FPGAs* foi detectada uma carência no mercado mundial de dispositivos com essas características. Dessa forma, o projeto ora proposto atende tanto às necessidades internas da Universidade de Brasília quanto o mercado global de sistemas reconfiguráveis.

[4] Em inglês, *Systems on Chip – SoC.*

Devido à falta de experiência no Brasil (e no GRACO-ENM[5]) no projeto de placas envolvendo *GPPs* e *FPGAs*, as tarefas foram encaminhadas pelo desenvolvimento de uma metodologia de projeto de sistemas com tais características.

Tendo em conta as circunstancias supracitadas, a proposta deste projeto foi desenvolver primeiro uma metodologia de elaboração de soluções em PCB e utilizá-la para elaborar uma plataforma para prototipagem com finalidades pedagógicas para pesquisa ou desenvolvimento de novos produtos. Isso possibilita ao projetista o desenvolvimento de sistemas embarcados, utilizando soluções conjuntas tanto em *software* (usando *GPPs*) como em *hardware* (usando *FPGAs*), ou hibridas. As aplicações mais evidentes para a plataforma são as dedicadas a tarefas de automação, especialmente aquelas com necessidade de controle por sistemas embarcados.

O projeto desenvolvido possui potencialidades pedagógicas as quais poderão ser exploradas nos diferentes cursos ministrados tanto na UnB como em outras universidades, no contexto de *computação reconfigurável* no âmbito dos *sistemas embarcados*. A robótica pedagógica já faz parte do currículo de diversos centros de pesquisa que por meio de protótipos permite o ensino de conceitos básicos de mecatrônica. *Kits* de desenvolvimento permitem que o professor promova soluções cotidianas com o aluno ressaltando a importância e a utilidade dos conceitos aprendidos pelos futuros profissionais (Wood, 2008).

O grande desafio foi obter uma plataforma operacional de modo a incluir, além das potencialidades inerentes ao *ARM* e ao *FPGA*, diversas interfases com o usuário ou outros sistemas. A interface com o usuário foi feita por meio de botões e *LEDs* ligados à cada processador. Adicionalmente, foi inserida uma memória de 1MB ligada ao *ARM7* e uma de 4MB ligada ao *FPGA*.

Aproveitando as funcionalidades inerentes ao *ARM*, interfaces *USB*, *JTAG*, RS232, SPI e I^2C também foram implementadas. Associadas ao *FPGA* foram disponibilizadas as comunicações Ethernet e *JTAG*, além de 16 vias de comunicação direta com o *ARM*. Todo o projeto foi concebido para que cada núcleo pudesse funcionar independente do outro. Toda a plataforma foi testada no Laboratório de Eletrônica GRACO, sendo verificado o adequado funcionamento dos núcleos e respectivos periféricos, independentemente um do outro bem como o funcionamento integrado de toda a plataforma.

Os princípios que nortearam desde o inicio dos trabalhos foram desempenho, segurança, baixo custo, robustez, flexibilidade e sustentabilidade.

[5] Grupo de Automação e Controle – Departamento de Engenharia Mecânica.

1.1 DESCRIÇÃO DO PROBLEMA

Diversos *kits* de desenvolvimento são comercializados tanto no mercado nacional como no internacional dispondo dos mais variados tipos de processadores. Todavia, na grande maioria das vezes, contam com apenas um núcleo de processamento (Bradley, 2010).

Inúmeros projetos dependem de um *hardware* com elevada capacidade de processamento de modo que, caso o *hardware* possua apenas um núcleo, pode ser que não suporte as requisições do projeto especialmente porque sistemas embarcados normalmente funcionam em baixa frequência (Bradley, 2010). Sistemas que envolvem o processamento de sinais como áudio e vídeo muitas vezes requerem um processador dedicado, por exemplo, à execução de cálculos e processamento propriamente dito (atuando como aceleradores de *hardware*) e outro voltado para a gerência do sistema.

Atualmente, é cada vez mais comum verificar a existência de dispositivos gerenciadores de multimídia voltados para conforto ambiental, mas por outro lado, é bem difícil um desenvolvedor ou aluno encontrar um único *hardware* agregando os requisitos mínimos necessários, especialmente envolvendo uma arquitetura reconfigurável (Kazmierkowski, 2011). Foi então identificou-se a necessidade de se desenvolver uma plataforma híbrida que cumpra esses requisitos com qualidade a um baixo custo a qual visa atender o desenvolvimento de projetos de sistemas embarcados para automação e computação ubíqua.

1.2 OBJETIVO

O objetivo desse trabalho é projetar, desenvolver e testar um sistema embarcado envolvendo computação reconfigurável para aplicações voltadas à computação ubíqua, mais precisamente, à domótica, sob um ambiente que possibilite a utilização de *software* livre para aplicações que envolvam monitoramento e controle eletrônico ou autonomia de sistemas. Cabe destacar que este trabalho é um subprojeto de um outro contemplado para financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq, cujo título é *Projeto de uma plataforma reconfigurável para o processamento de áudio, vídeo e navegação em ambientes de computação ubíqua*, número 133734/2009-9

Por meio de análise comparativa com outros produtos disponíveis no mercado, e buscando suprir as necessidades dos laboratórios de ciências mecatrônicas e ciências da computação, na UnB, o presente trabalho propõe-se a desenvolver esta solução, em *hardware* reconfigurável, de forma a servir de protótipo de desenvolvimento de outros projetos ou produtos dependentes de uma plataforma de sistema embarcado. Esta solução busca atender tanto o ambiente acadêmico como o desenvolvimentista industrial com a junção de duas tecnologias de desenvolvimento, uma mais tradicional, *GPP*, e outra mais atual, *FPGA*. O grande desafio foi a própria criação deste sistema em que cada núcleo possa funcionar separadamente ou em conjunto, dependendo da necessidade do projetista, garantindo os requisitos básicos do projeto: qualidade, segurança, flexibilidade, portabilidade, baixo custo e sustentabilidade ambiental.

Por mais que ao longo do trabalho tenham surgido plataformas com características semelhantes no mercado, a simples compra da solução priva toda a equipe dos conhecimentos de fabricação e desenvolvimento. A posse desse conhecimento é de fundamental importância não somente para a Universidade de Brasília como também para o Brasil pois representa mais independência para evolução tecnológica dos processo produtivos.

A metodologia desenvolvida e utilizada deverá servir de subsidio para o desenvolvimento de outros tipos de plataformas ou sistemas embarcados, que possam ser aplicadas de maneira eficiente em áreas de mecatrônica, como automação predial/residencial (domótica), computação ubíqua, visão computacional, controle aplicado a robótica , entre outras.

1.2.1 OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste trabalho são:

1. Levantamento de requisitos do sistema.
2. Selecionar dois cernes de processamento para compor os núcleos de uma plataforma de processamento de dados/sinais com disponibilidade para serviços de navegação, automação, aplicações em robótica e controle, visão computacional, projetos de *hardware*/*software*, domótica, computação ubíqua, entre outros;
3. Desenvolvimento de uma metodologia para projeto de *hardware* no contexto de sistemas embarcados mistos (*GPPs* e *FPGAs*), que venha fornecer elementos práticos e experiência à equipe do GRACO-UnB;

4. Seleção de periféricos para suporte a cada um dos núcleos selecionados;
5. Elaboração dos circuitos eletrônicos constituintes de cada núcleo separadamente;
6. Encontrar um meio adequando de estabelecer a comunicação entre o *GPP* e o *FPGA*;
7. Seleção dos componentes, tendo como referência os requisitos de qualidade, flexibilidade, desempenho, baixo custo e sustentabilidade ambiental;
8. Elaboração do *layout* da placa de circuito impresso;
9. Confecção da placa de circuito impresso conforme *layout* elaborado;
10. Montagem dos componentes na placa de circuito impresso;
11. Execução de testes de *hardware* e avaliação de desempenho;
12. Execução de testes com *softwares* e avaliação de desempenho;
13. Avaliação de potencialidades da plataforma reconfigurável e sua utilização como sistema embarcado;
14. Sugestão de possíveis melhorias para projetos derivados.

1.3 JUSTIFICATIVA

Sistemas reconfiguráveis estão deixando de ser mais uma opção de projeto e estão se tornando uma necessidade para projetos embarcados (Hartenstein, 2006). Eles possuem vantagens comparativas em relação a plataformas baseadas unicamente em *GPP*, especialmente em termos de consumo de energia, flexibilidade e performance (Correia, 2007).

O domínio dos sistemas computacionais tradicionais (baseados na arquitetura de von Neumann) gradualmente está abrindo espaço para a computação reconfigurável a qual promete trazer profundas mudanças práticas tanto para a computação cientifica quanto para sistemas ubíquos embarcados. Ela tende a ser a ferramenta para romper as barreiras entre a computação de alta performance e o cotidiano das pessoas (Hartenstein, 2006).

Atualmente, a fronteira tecnológica dos sistemas ubíquos depende do avanço da computação reconfigurável (Edwards; Green, 2003). Seguindo as tendências de futuro da eletrônica básica espera-se que a plataforma

elaborada neste trabalho sirva de subsidio para o desenvolvimento da eletrônica nacional e ainda possa dar suporte a projetos de automação ubíqua.

A principal justificativa para o início do desenvolvimento deste projeto é a reduzida oferta de uma plataforma semelhante no mercado, envolvendo dois núcleos de processamento em que um deles seja reconfigurável, assim como interfases apropriadas para trabalhos de automação, onde possam ser aplicados os conceitos de computação embarcada.

Finalmente, há uma tendência atual para o incremento da quantidade de *software* direcionada para sistemas embarcados. Conforme a *Lei de Rammig*, a quantidade de *softwares* desenvolvidos para sistemas embarcados é dobrado a cada dez meses (Rammig, [S.d.]). Além disso, estima-se que 90% de todos os *software* produzidos para sistemas embarcados atualmente são voltados para o uso em *FPGA*s (Hartenstein, 2006). Tendo em conta que o uso de *FPGAs* em sistemas embarcados tende a crescer nos próximos anos e que os *GPPs* dominaram por anos o mercado de sistemas embarcados, chega-se a uma solução pela utilização de sistemas híbridos (*GPP* + *FPGAs*), seja em plataformas de desenvolvimento ou em produtos finais.

Essa junção de arquiteturas envolvendo soluções em *software* e *hardware* no mesmo sistema denomina-se *co-design* e vem sendo muito utilizado atualmente pois viabiliza extrair o que cada tecnologia tem de melhor para a construção da solução. De maneira rápida, pode-se afirmar que os *GPPs* apresentam vantagens de serem mais baratos que os *FPGAs* mas por outro lado os *FPGAs* consomem menos energia que os *GPPs*. A ilustra a relação mais vantajosa de cada arquitetura.

Tabela 1 – Vantagens comparativas entre GPPs e FPGAs.

	GPPs	***FPGA***
Vantagem	$\frac{Desempenho}{\$}$	$\frac{Desempenho}{energia}$

1.4 ESCOPO DO PROJETO

O escopo deste projeto restringe-se à proposta de uma metodologia de desenvolvimento e teste de uma plataforma com dois núcleos de processamento, em que um deles seja reconfigurável. Esta plataforma tem como principal público alvo engenheiros projetistas e estudantes interessados em conhecer e desenvolver sistemas de controle em ambiente reconfigurável.

A plataforma tem as seguintes características: (a) não foi projetada para ser utilizada em ambientes industriais com elevados níveis de radiação ou interferência eletromagnética; (b) não deve ser utilizada em ambientes com temperatura abaixo de 0ºC e nem acima de 50ºC (limites de funcionamento dos componentes utilizados); (c) não deve ser utilizada para fins militares, paramilitares ou que envolva qualquer forma de tecnologias nucleares, aeroespaciais ou armamentistas, conforme termo assinado para a importação dos componentes utilizados; (d) plataforma deve ser alimentada com tensão contínua entre 9 e 12V e corrente mínima de 1A; (e) a plataforma possui as dimensões de 160 X 150 X 20mm.

No que se refere ao *layout* da placa, ela possui quatro camadas, sendo as intermediárias dedicadas à alimentação – GND e VCC -, e as de superfície para trilhas de sinais e fixação de componentes.

Finalmente, o projeto não envolve o desenvolvimento de *software*s ou sistemas de controle, tão somente a parte física da plataforma. Os *softwares* que foram desenvolvidos restringem-se apenas à verificação e teste do funcionamento dos componentes críticos da plataforma.

1.5 CONTRIBUIÇÕES DO PROJETO

O produto resultante deste trabalho pode contribuir com a formação de engenheiros durante os cursos de Engenharia Mecatrônica, Controle e Automação, Ciência da Computação ou outros relacionados à computação embarcada reconfigurável. Adicionalmente, o mesmo agrega experiência ao grupo de eletrônica do GRACO, na elaboração de soluções para diferentes tarefas.

Finalmente, a avaliação de desempenho de uma plataforma como esta pode subsidiar a elaboração de novas plataformas, utilizando outros dispositivos, ou novas soluções não apenas no ambiente acadêmico mas também no mercado.

1.6 ORGANIZAÇÃO DO TEXTO

Este documento foi organizado buscando proporcionar ao leitor uma leitura agradável com a apresentação das tecnologias utilizadas no projeto e desenvolvimento da plataforma reconfigurável de controle. Para apresentar

estas informações de forma organizada e metódica, este trabalho foi distribuído em sete capítulos e cinco anexos.

No CAPÍTULO 1 é feita uma breve introdução ao tema, seguida da apresentação dos objetivo e principais aspectos orientadores do trabalho. Em seguida são dadas as principais justificativas para esta escolha e finalizando com a delimitação do escopo do sistema.

No CAPÍTULO 2 é apresentada a fundamentação teórica abordada na elaboração do projeto. São apresentados os conceitos de sistemas embarcados, sua evolução em sistemas ubíquos e ainda os princípios de utilização e características dos processadores voltados à computação reconfigurável.

A seguir, no CAPÍTULO 3, é detalhada a metodologia elaborada tanto para o desenvolvimento do projeto como para a fase de testes. Neste capítulo, é mostrado, em detalhes, os processos que foram seguidos desde as fases de concepção do projeto, desenvolvimento de circuitos, confecção da placa de circuito impresso (PCI) até os testes da plataforma, depois de montada.

O CAPÍTULO 4 é dedicado ao detalhamento dos principais pontos concernentes às fases de projeto e desenvolvimento da plataforma. Nele é detalhado cada módulo implementado, função de cada dispositivo bem como as entradas e saídas do sistema. É neste capítulo que são apresentados todos os circuitos eletrônicos utilizados.

A seguir, no CAPÍTULO 5, são apresentados os testes realizados e os resultados obtidos. Neste capítulo, detalha-se a sequencia de passos adotados para a execução dos testes prezando pela integridade de toda a plataforma. São explicados os motivos, por exemplo, de os testes serem conduzidos em cada módulo separadamente e apenas depois de verificado seu correto funcionamento que é feita a integração.

As conclusões de todo o trabalho são apresentadas no CAPÍTULO 6, bem como as sugestões de melhoria para possíveis evoluções do projeto.

Adicionalmente, no APÊNDICE A, são apresentados os circuitos elétricos desenvolvidos pelas ferramentas *CAD/EDA* e utilizados na concepção da plataforma. No APÊNDICE B é feita uma apresentação da plataforma como resultado final. Por meio de fotos e esquemáticos, são destacados os componentes utilizados e topologia adotada. É feita também a descrição das funcionalidades envolvidas no chaveamento por *jumpers*, conectores e seletores da plataforma.

O APÊNDICE C destina-se a apresentar, em detalhes, o *layout* do projeto PCB da plataforma, e ainda as quatro camadas (*layers*) condutoras separadamente.

Já no APÊNDICE D, é apresentada a lista de todos os componentes necessários para a montagem de cada unidade da plataforma reconfigurável.

O APÊNDICE E traz, em detalhes, todas as ferramentas utilizadas para a programação bem como os procedimentos e configurações necessárias em cada uma delas. no APÊNDICE F estão disponibilizadas todas as linhas de código de programação utilizadas tanto no *ARM* quanto no *FPGA*. Por fim, no APÊNDICE G estão disponibilizados os circuitos constituintes da segunda versão da plataforma.

CAPÍTULO 2
FUNDAMENTAÇÃO TEÓRICA

A seguir serão apresentadas as fundamentações teóricas mais relevantes necessárias durante a fase de desenvolvimento do projeto.

2.1 SISTEMAS EMBARCADOS

A ênfase das pesquisas em sistemas eletrônicos há algum tempo já se voltou para um mercado muito mais vasto que são os sistemas embarcados. Eles são definidos como sistemas computacionais para uso específico ou dedicados (Saint-Jean *et al.*, 2007) também conhecidos como *sistemas embutidos*, possuem basicamente as mesmas partes de um computador mas a especificidade de suas tarefas faz com que, muitas vezes, não sejam nem usados nem percebidos como tal (Srivastava *et al.*, 1998). São exemplos típicos destes sistemas: câmeras digitais, roteadores de rede IP, hodômetros, pilotos automáticos e sistemas de controle em geral.

Os projetos de sistemas embarcados podem apresentar uma grande flexibilidade não apenas do ponto de vista da programação, *software*, mas também em relação à parte física, *hardware*, especialmente quando implementados em sistemas reconfiguráveis. *Hardware* específico ou dedicado, normalmente é separado para tarefas que exigem alto poder de processamento, assim as demais funcionalidades são implementadas por meio de *software* (Wei *et al.*, 2008).

Sistemas embarcados, há certo tempo, vem ganhando força pelos requisitos de economia de energia, portabilidade, complexidade de processamento e baixo custo (Saint-Jean *et al.*, 2007). Atualmente, os sistemas embarcados são a base tecnológica da computação ubíqua (Sudha *et al.*, 2007).

2.2 CONTROLE E AUTOMAÇÃO NO CONTEXTO DE SISTEMAS EMBARCADOS

A economia mundial foi marcada no século XVIII pela invenção da máquina a vapor; no fim do século X, os propulsores da nova revolução do desenvolvimento são a tecnologia (representada pela informática e pelo aperfeiçoamento dos transportes e das comunicações) e a globalização (Rosário, 2005).

Provavelmente os dias atuais entrarão para a história como o período da *moderna revolução industrial*, numa analogia com o período inicial da industrialização, no século, XVIII, quando o homem passou a controlar os sistemas de potência. Na *moderna revolução industrial* o homem conseguiu o controle sobre os sistemas de informação porque a informação atualmente é o ativo mais valioso para qualquer organismo (Duarte, 1997).

Não por acaso é observado um crescimento da internet, que se dá justamente em um momento em que as fronteiras entre os estados e os mercados estão se diluindo e a tecnologia, principalmente de comunicação e informação, constitui-se um forte elemento de transformação (Costa, 2002).

Investimentos em tecnologia privilegiam a inovação como vantagem competitiva. As estratégias empresariais são definidas com base na identificação de oportunidades, onde a competição é fundamental, sendo baseada em avanços desenvolvidos em centros de pesquisa, onde os custos do processo e a cadeia produtiva têm papel de destaque (Utterback, 1996). Dessa forma, os investimentos em pesquisa e desenvolvimento passaram a fazer parte da agenda, não só das empresas na fronteira tecnológica mas também daquelas que buscam um lugar de relevância no mercado.

Nesse contexto, a automação e controle ganham força, pois possibilitam a utilização de tecnologias de ponta, a um baixo custo, e ainda conjuga propriedades multidisciplinares proporcionando a redução do *Time to Market* – TTM[6] dos produtos (Westine, 1986).

A automação e controle (também conhecida como mecatrônica) é uma das áreas recentes da engenharia e caracteriza-se por ser interdisciplinar por natureza, tendo adquirido vida própria à medida que a automação industrial começou a ser difundida como uma solução para o aumento de produtividade (Ibrahim, 2003). Ela foi criada ao longo da vida profissional de uma geração de

---

[6] *Time to Market* é o tempo de colocação do produto no mercado, sem que afete o nível de demanda e oferta do mesmo. É o tempo de projeto e concepção de um produto ou serviço até a disposição deste produto para o consumidor final.

engenheiros que ainda está ativa, integrando conhecimentos de diversas áreas tradicionais, como mecânica, elétrica e a computação. Ela prima por ser rica em aspectos tecnológicos e inovadores. Ao engenheiro eletricista, especialmente ao eletrônico, com visão voltada para o desenvolvimento de placas de circuito impresso, associa-se o mecânico, quando é necessário o conhecimento do movimento de corpos no espaço e da resistência estrutural do sistema, sua flexibilidade e as consequentes vibrações. A linguagem comum entre esses componentes é o campo do engenheiro mecatrônico e, finalmente, o resultado de toda essa soma é a vida moderna, em que todos esses novos dispositivos opto-eletro-mecânicos já estão incorporados ao dia-a-dia das pessoas (Rosário, 2005).

A mecatrônica pode, portanto, ser compreendida como uma filosofia relacionada à aplicação combinada de conhecimentos de áreas tradicionais, como a engenharia mecânica, a eletrônica, controle e a computação, de forma integrada e complementar (ATUAL, [S.d.]).

Outros autores apresentam a Mecatrônica como uma linha de pensamento que combina os conceitos de mecânica, eletrônica e computação, no desenvolvimento de produtos que combinem conceitos de resistência dos materiais, comportamento térmico, mecanismos, sensores, atuadores, controle, lógica, entre outros (Adamowski e Furukawa, 2001). Neste sentido, Schweitzer (1996) apresenta mecatrônica como uma área interdisciplinar que envolve engenharia mecânica, engenharia elétrica e ciências da computação. Por outro lado, Acar (1996) a define como sendo a integração da microeletrônica, computação e controle de sistemas mecânicos, para a solução de projeto e produtos com inteligência e flexibilidade.

Assim, essa formação interdisciplinar forma a estrutura básica do engenheiro mecatrônico, ou de controle e automação, cuja atividade mais tradicional é a elaboração e desenvolvimento de sistemas embarcados.

2.3 DOMÓTICA, SISTEMAS UBIQUOS E A RELAÇÃO COM SISTEMAS EMBARCADOS

A *domótica* está relacionada com a implementação de soluções integradas de automação predial e residencial, ou seja, diz respeito à computação ubíqua aplicada à gestão de qualquer recurso habitacional (Aiello; Dustdar, 2008). O termo *domótica* resulta da junção da palavra latina *domus*, que significa casa, com robótica.

Por outro lado, *sistemas ubíquos*, também conhecidos por computação ubíqua, diz respeito ao suporte computacional contínuo e permanente ao ser humano (Satyanarayanan, 2001). A mobilidade aliada à difusão da comunicação sem fio (*wireless*) permitiu aos sistemas computacionais serem conscientes do contexto ambiental e interagirem com a sociedade (Buchanan, 2001).

Durante as últimas décadas, observou-se a vigorosa entrada de equipamentos eletrônicos no dia a dia das pessoas. Eles se fazem cada vez mais presentes nos mais diversos setores (Kunito *et al.*, 2006).

Grande parte desse crescimento pode ser imputado às *Tecnologias da Informação e Comunicação* (TICs) que são grandes demandantes de equipamentos eletrônicos e vem contribuindo para a massificação do uso desses produtos. TICs são consideradas como uma das principais forças propulsoras do aumento de equipamentos eletrônicos, pois, essencialmente, dependem de um conjunto de recursos tecnológicos integrados entre si que proporcionam, por meio de *hardware* e *software,* a produção, o *arma*zenamento e o compartilhamento de informações (Augustin *et al.*, 2004).

Então, a rigor, pode-se encontrar a eletrônica (sistema embarcados) desde os equipamentos básicos como relógio, termômetro, sensor de presença, passando pelas telecomunicações como celulares, *smartphones*, roteadores, *access point*, TV's e chegar até níveis mais sofisticados como satélites e suas centenas de subsistemas. Grande parte deles interconectados e interdependentes, de modo a oferecer soluções eficientes de telecomunicação e informação ao usuário (Castells; Majer; Gerhardt, 2000).

De fato, a eletrônica tem avançado tanto que é possível, e na verdade já vem ocorrendo, a integração *homem-máquina* de uma maneira muito sutil. São os denominados sistemas ubíquos que por meio de uma série de equipamentos de uso específicos trazem a informática ao cotidiano das pessoas. O termo e sua definição são atribuídos a Mark Weiser, que o propôs na década de 90 (Weiser, 1993).

Essa interação não é para ser invisível, mas sim natural, de uma forma que as pessoas percebam que estão interagindo com um dispositivo, mas tão naturalmente quanto uma conversa com alguém (Vukosavljev *et al.*, 2011). Neste contexto, alguns autores afirmam até que computadores pessoais e estações de trabalho se tornarão obsoletos pois o acesso computacional à informações estará disponível em todos os lugares (Weiser, 1993).

Essa vasta possibilidade de interação *homem-máquina* é o combustível para o desenvolvimento da eletrônica para sistemas embarcados, orientados a uma ou várias aplicações. Nesse ponto, percebe-se a dependência que os sistemas ubíquos e domóticos têm dos sistemas embarcados ou dedicados. A

onipresença, portabilidade e onisciência dos sistemas só são possíveis por meio da microeletrônica embarcada (Weiser, 1991).

Fica fácil perceber o potencial de integração que têm esses sistema pois exigem características como desempenho, baixo consumo, flexibilidade, facilidade de implementação, etc. (Bonino; Corno, 2011). A plataforma ora desenvolvida apresenta-se como possível solução para grande parte desses sistemas pois agrega as características mais relevantes dos sistemas embarcados.

2.4 NÚCLEOS DE PROCESSAMENTO

Os núcleos de processamento para sistemas embarcados são, sem dúvida, os principais componentes físicos dos sistemas embarcados. Uma vez definido o processador, conforme os requisitos mínimos do projeto, os demais circuitos e componentes, normalmente, são apenas acessórios para a composição da solução (Talla; John, 2003). Eles podem ser classificados como de uso geral – *GPP,* quando são construídos para atenderem a diversos tipos de aplicações e funcionam pela execução de um programa, ou de propósitos específicos, *ASICs* (Debes *et al.*, 2002). Quando de propósito geral, como o próprio nome já sugere, possui a característica de disponibilizar diversas funções implementadas em *hardware,* as quais, na maioria dos projetos, nem todas são utilizadas (Ghasemi *et al.*, 2003). Então, tarefas como, buscar, decodificar e executar instruções pode sobrecarregar este tipo de processador.

Por outro lado, há também circuitos integrados projetados especificamente para uma ou um conjunto de aplicações (*ASIC - Application Specific Integrated Circuit*) que têm desempenho, em execução de tarefas, superior aos de propósitos gerais vez que são dedicados a uma ou algumas poucas aplicações (Burge; Grout; Dorey, 1994)(Gregson *et al.*, 1989).

Já soluções construídas em *hardware* têm a vantagem do paralelismo inerente ao circuito eletrônico, onde muitos fluxos de dados podem coexistir (Compton; Hauck, 2002). Abordagens em *hardware* apresentam índices de aceleração (*speed-up*) superiores quando comparadas às abordagens em *software*. O fato de os microprocessadores executarem sequencialmente suas tarefas, por seguirem o modelo de von Neumann, faz com que os respectivos tempos de processamento não sejam aceitáveis para muitas aplicações (Dido *et al.*, 2002; Pimentel; Le-Huy, 2000). Por outro lado, a principal desvantagem de projetos baseados em *hardware* fixo é a perda de flexibilidade durante a fase de desenvolvimento, pois uma vez construído o dispositivo físico, não há como alterá-lo. Essa dificuldade torna-se a principal vantagem ao utilizar computação

reconfigurável, *FPGA*, pois ela permite essa alteração em *hardware* (mesmo em tempo de execução, usando técnicas de reconfiguração dinâmica e parcial), durante o uso do dispositivo.

Diante das várias possibilidades disponíveis no mercado, cabe ao desenvolvedor fazer a melhor escolha de modo a atender as especificações do projeto.

Em sistemas embarcados é comum a utilização de microcontroladores em vez de processadores comuns (Pereira; Carro, 2007). O microcontrolador é um tipo de processador que além de possuírem os componentes lógicos e aritméticos, eles ainda integram elementos adicionais em sua estrutura interna, tais como memória de leitura e escrita para armazenamento de dados, memória somente de leitura para armazenamento de programas, *EEPROM* para armazenamento permanente de dados, dispositivos periféricos como conversores analógico/digitais (*ADC*), conversores digitais/analógicos (*DAC*), entrada para interrupção externa, saída *PWM*[7] (moduladores por largura de pulso) além das interfaces de entrada e saída de dados (Ro; Gaudiot, 2009).

2.5 MICROCONTROLADORES - *ARM7* (UTILIZADOS NA PLATAFORMA)

O *ARM7* é um processador RISC de 32 bits, *pipeline* de três estágios e arquitetura de registradores *load*/*store*. Neste contexto, o processador estará sempre ocupado ao executar três instruções em diferentes estágios. Enquanto busca a primeira, decodifica a segunda e executa a terceira (Silvestre; Bachiega, 2007, p. 7). Essa arquitetura, chamada *pipeline,* é bem simples e evita os conflitos entre estágios de arquiteturas mais avançadas. O processador *ARM7* possui sete diferentes modos de execução. Partes desses modos são usados para tratamento de exceções e de interrupções. Um sistema operacional pode utilizar os modos privilegiados para executar códigos de sistema e deixar as aplicações restritas no modo usuário (Yiu, 2007).

Tabela 2 – Modos de operação do *ARM7*.

Item	***Modo***	***Descrição***
1	Usuário	Execução normal de programas
2	Sistema	Executa rotinas privilegiadas do sistema operacional
3	Supervisor	Modo protegido para o sistema operacional
4	IRQ	Tratamento de interrupções comuns

[7] Em inglês: *Pulse Width Modulation*

5	FIQ	Tratamento de interrupções rápidas
6	Abort	Usado para implementar memória virtual ou proteção de Memória
7	Indefinido	Suporta a emulação em *software* de co-processadores

Ainda segundo Silvestre e Bachiega (2007) todos os perfis da arquitetura *ARM7* implementam a tecnologia *Thumb*® de compressão de código. O modo *Thumb* é um modo especial que transforma o processador *ARM* em um *CORE* de 16 bits, aumentando assim o espaço disponível na memória de programa. Essa mudança pode ser feita em tempo de execução; dessa forma o *ARM7* possui dois conjuntos de instruções um de 32 bits e outro de 16 bits. O segundo conjunto é mais simples que o primeiro, mas continua processando dados em 32 bits. Todos os registradores no modo *Thumb* continuam sendo acessados como 32 bits. A arquitetura também inclui as extensões da tecnologia *NEON*™ para aumentar o DSP (*Digital Signal Processor*) e o processamento digital em até 400%. Adicionalmente, ela oferece melhor suporte à operação com ponto flutuante, no endereçamento que a nova geração de gráficos 3D e jogo requisitam, assim como aplicações tradicionais de controle embarcado.

Essa família se consagrou no mercado por ter sido utilizada em vários dispositivos de utilização em massa, em destaque: *Apple iPod, Nintendo Ds e Game Boy Advance*, vários celulares Nokia, Lego *Mindstorms* NXT, SEGA *Dreamcast* (processamento de áudio) (Wikipedia contributors, 2012). Sua popularidade pode ser devido a estudos que indicam que um *GPP*, como o *ARM7*, apresenta melhores resultados, se comparados com *FPGAs*, em desempenho por dólar investido (Hamada *et al.*, 2009).

Os CIs *ARM* não são produzidos por uma única empresa, na verdade, eles são licenciados e produzidos por diversos fabricantes. A *ARM* Ltd. é a empresa responsável apenas pelo desenvolvimento do *ISA* (sistema de informações da arquitetura), detendo assim, todos os direitos sobre sua arquitetura. Com isso, ela pode comercializar diversos tipos de licenças para a fabricação dos CIs (Yiu, 2010).

No desenvolvimento deste trabalho, optou-se por utilizar os CIs do fabricante NXP pelo fato de haver, no laboratório do GRACO, alguns *kits* de desenvolvimento utilizando o *ARM7* da família LPC2000 desse fabricante, assim como toda a documentação de apoio. Além disso, apresenta custos razoáveis, o fabricante agrega segurança ao projeto pela solidez e tradição da

empresa além de ser um produto ambientalmente sustentável por cumprir a regra RoHS[8].

Ao ser analisada a família LPC2000 da NXP percebeu-se que a escolha do LPC2146 seria ainda justificável pela relação custo benefício, tendo em conta algumas características, descritas a seguir[9]:

a) 32-bit *ARM7*TDMI-S Microcontrolador no pequeno encapsulamento LQFP64.

b) 40 kB de memória RAM estática *on-chip* e 256 kB de memória flash programável, *on-chip*. Amplitude de 128 bit que possibilita operações em 60 MHz.

c) *In-System/In-Application Programming* (ISP/IAP) via *software boot-loader on-chip*. Limpar a memória *flash* em setores ou completamente em 400ms e programação de 256 *bytes* em 1ms.

d) Interfaces embarcadas CE RT e Trace oferecem deputação (*debugging*) em tempo real *software RealMonitor on-chip* e alta velocidade de rastreamento de execução das instruções.

e) Dispositivo controlador de *USB* 2.0 de velocidade total com 2 kB de *endpoint RAM*. Adicionalmente, os LPC2146/8 contam ainda com 8 kB de RAM *on-chip* acessível para *USB* por *DMA*.

f) Conversores *A/D* de 10-bits proporcionam um total de 6/14 entradas analógicas, com o tempo de conversão abaixo de 2,44 µs por canal.

g) Conversores singulares *D/A* de 10-bit proporcionam saídas analógicas variáveis.

h) Dois temporizadores/contadores de eventos externos de 32-bit (com quatro canais de captura e quatro canais de comparação cada), unidade *PWM* (seis saídas) e *watchdog*.

i) *Clock* de tempo real de baixa potência com entrada independente de alimentação e entrada dedicada de *clock* de 32 kHz.

j) Múltiplas interfaces seriais incluindo duas *UARTs* (16C550), duas I^2C rápidas (400 kbit/s), SPI e SSP com *buffering* e recursos de variação de compressão de dados.

---

[8] *RoHS (Restriction of Certain Hazardous Substances*, Restrição de Certas Substâncias Perigosas) é uma diretiva europeia (não é lei ainda) que proíbe que certas substâncias perigosas sejam usadas em processos de fabricação de produtos: cádmio (Cd), mercúrio (Hg), cromo hexavalente (Cr(VI)), bifenilos polibromados (PBBs), éteres difenil-polibromados (PBDEs) e chumbo (Pb).

[9] User manual LPC214x - Chapter 1: Introductory Information - Rev. 3 — 4 October 2010. Disponível em: www.nxp.com. Acesso em: Junho 2011.

k) Controlador de interrupção vetorizada com priorização de configurações e endereço vetorial.

l) Até 45V de tolerância nas entradas e saídas de propósito geral de 5V no pequeno encapsulamento LQFP64.

m) Nove pinos externos de interrupção disponíveis (transição ou nível).

n) Disponibilidade máxima de 60 MHz de *clock* da CPU de PLL (*phase-locked loop*) programável *on-chip* com tempo de configuração de 100 μs.

o) Oscilador integrado *on-chip* opera com cristal externo na faixa de 1MHz a 30MHz e com oscilador externo em até 50MHz.

p) Modos de economia de energia incluem *Idle* e *Power-down*.

q) Funções periféricas com liga/desliga individuais bem como os *clocks* periféricos escalonados para otimizações adicionais de energia.

r) *Wake-up* do processador a partir da função *Power-down* via interrupção externa, *USB*, *Brown-Out Detect* (BOD) ou *Real-Time Clock* (RTC).

s) Fonte única de suprimento de energia com circuitos *Power-On Reset* (POR) e *Brown-Out Detection* (BOD).

t) A CPU opera dentro da faixa de voltagem de 3.0V to 3.6V (3.3 V ± 10 %) com entrada e saídas tolerantes até 5V GPIO (*general purpose input output*).

2.6 A COMPUTAÇÃO RECONFIGURÁVEL BASEADA EM *FPGAs*

Nos últimos anos, projetos de *hardware* dedicado têm sofrido rápida evolução, assim como as tecnologias de *software* e *hardware* (Kim; Park; Tak, 2009). É muito grande a quantidade de novos microprocessadores, interfaces de comunicação, interfaces de potência, sensores, compiladores, sistemas operacionais, assim como sistemas de desenvolvimento fornecidos aos mercado a cada ano. Em virtude da acelerada evolução tecnológica a ideia de utilizar estruturas abertas e reconfiguráveis, capazes de adaptarem-se às novas demandas, torna-se muito atraente e se constitui pré-requisito na consideração de um projeto de sistema embarcado (Edwards; Green, 2003).

Sistemas de desenvolvimento baseados em computação reconfigurável (sistemas de *hardware* reconfigurável) apresentam características adequadas para auxiliar na execução dessa classe de projeto. Apresentam ainda, entre outras vantagens, baixo consumo de energia, alta velocidade de execução de operações, capacidade de integração, flexibilidade e operação modular (Hartenstein, 2007; Rosário, 2005).

Em geral, sistemas reconfiguráveis são aqueles que, mediante a substituição de parte de seu *software* ou *hardware*, apresentam a característica de adaptar-se a tarefas específicas. Eles têm por objetivos obter alto desempenho com baixo consumo de energia, sendo uma alternativa às máquinas de von Neumann implementadas pelos sistemas com microprocessadores (Hartenstein, 2006). Conforme pesquisas, o imenso consumo de energia tem sido considerado um dos obstáculos mais severos para se chegar aos supercomputadores *petaflop*[10], utilizando tecnologias clássicas (Hartenstein, 2007).

Atualmente, sistemas de *software* reconfigurável são utilizados nos mais diversos dispositivos (Shima, 2005). Nesses sistemas, a mudança de uma *ROM* leva à reconfiguração de funções responsáveis por sua operação (Miyazaki, 1998). Os *FPGAs* (*Field Programmable Gate Array*) podem ser definidos como sistemas de *hardware* reconfigurável e surgiram no inicio da década de 90. O advento de dispositivos como os *FPGAs* mudou o ponto de equilíbrio entre compromisso, flexibilidade e desempenho.

Com os *FPGAs*, os requisitos de performance são mantidos com o aumento de flexibilidade (Kalra, 2001; (Hartenstein, 2006; Hubner *et al.*, 2010). Além disso, estudos indicam que *FPGAs* apresentam melhores resultados, se comparados com *GPPs*, em desempenho por unidade de energia consumida (Hamada *et al.*, 2009).

Comparados aos sistemas de *software* reconfigurável, os sistemas de *hardware* reconfigurável apresentam maior potencial no que diz respeito à performance e adaptabilidade.

Outros nomes também são dados aos sistemas de *hardware* reconfigurável: CCM (*custom computing machine – sistemas computacionais customizados*) e FCCM (*based custom computing machine – hardware* reconfigurável baseado em sistemas computacionais customizados). O termo *computação reconfigurável* ou *lógica reconfigurável* também está associado a sistemas de *hardware* reconfigurável (Rosário, 2005).

A computação reconfigurável tem por objetivo suprir a lacuna entre a solução por *software* e a solução por *hardware* atingindo desempenhos muito superiores aos obtidos por *software*, mas com flexibilidade muito maior que a oferecida por uma solução exclusiva por *hardware* (Chen; Chen; Chen, 2000; Compton; Hauck, 2002; Coric *et al.*, 2005).

No caso de sistemas embarcados, eles possuem elevado dinamismo tecnológico, ou seja, estão sujeitos a grandes variações tecnológicas em curto

---

[10] 10^{15} operações de ponto flutuante por segundo.

espaço de tempo, seja por demandas por novas tarefas e performances, seja por disponibilidade de novas tecnologias de sensores e atuadores. Restrições de recursos em sistemas embarcados acentuam a necessidade de novas ideias de projeto. Neste contexto, é comum a utilização de blocos de *software* funcionalmente testados para acelerar o tempo de projeto de um sistema de uma interface de programa aplicada (API – *application program interface*). Existem blocos funcionais que podem assumir a mesma função no caso do *hardware*; eles podem ser combinados com outros circuitos para implementar um determinado algoritmo. Trata-se, neste caso, de um aspecto modular que facilita a manutenção e a modernização de projetos (Compton; Hauck, 2002; Coric *et al.*, 2005; Renner *et al.*, 2000).

A utilização de aplicativos desenvolvidos por meio de sistemas proprietários do tipo *IP-cores* (*intellectual property cores*), disponíveis para serem integrados em outras aplicações, permite a integração de soluções já desenvolvidas por diversos fornecedores para minimizar o tempo de projeto. Barramento PCI, interfaces de comunicação e funções de processamento de sinal (como a implementação de algoritmos para transformada rápida de Fourier FFT – *Fast Fourier Transform*), codificação e decodificação de imagens digitais e mesmo microprocessadores e DSPs (*Digital Signal Processors*) são exemplos de *IP-cores* disponíveis (Ito; Carro, 2000; Kean, 2002).

Outros dispositivos, sem ser do tipo *FPGAs*, permitem a execução de algoritmos diretamente em *hardware*. Neste sentido, é possível classificá-los como dispositivos lógicos programáveis ou PLDs (*Programmable Logic Devices*), dispositivos lógicos programáveis complexos ou CPLDs (*Complex Programmable Logic Devices*), *FPGAs* (*Field Programmable Gate Arrays*) e sistemas dinâmicos de *hardware* reconfiguráveis ou *DFPGAs* (*Dinamically Field Programmable Gate Arrays*) (Miyazaki, 1998). Com esses dispositivos programáveis é possível executar os algoritmos explorando o paralelismo inerente da solução por *hardware*, muito mais rapidamente do que se eles fossem executados de forma sequencial por microcontroladores ou por *DSPs*, sujeitos ao modelo de von Neumann. Por outro lado, PLDs e *FPGAs* apresentam diferenças quanto à aplicação e quanto à estrutura interna. Neste contexto, *FPGAs* são geralmente aplicados em sistemas com menor sensibilidade ao custo e de maior complexidade (Pimentel; Le-Huy, 2000).

Paralelamente ao desenvolvimento de dispositivos que permitiram a implementação computacional de dispositivos reconfiguráveis, também foram desenvolvidas ferramentas de projeto, depuração, simulação e testes (Su *et al.*, 2006). Tais ambientes possibilitam a criação de módulos desenvolvidos com linguagens de alto nível de abstração, denominadas *linguagens de descrição de hardware* (HDLs – *Hardware Description Languges*), a exemplo da VHDL [*VHSIC (*Very High Speed Integrated Circuits*) hardware descriptive language*] e

da AHDL (*Altera hardware descriptive language*) (Altera, 2002; Navabi & Day, 1991; Pimentel & Le-Huy, 2000).

A grande aceitação do mercado pelos *FPGAs* também alcançou a indústria de *EDA* (*Electronic Design Automation)* e CAD (*Computer-aided design)*. Assim como aconteceu para os *GPPs*, todas as grandes empresas hoje fazem esforços substanciais para oferecer uma variedade de ferramentas de desenvolvimento e ambientes destinados ao desenvolvimento de projetos utilizando *FPGAs* (Hartenstein, 2006).

Do mesmo modo, é possível implementar blocos com representações de mais baixo nível de abstração, a exemplo dos esquemáticos (linguagem gráfica). A integração de blocos criados com diferentes linguagens permite uma concepção bastante flexível de projetos, com fácil interação da equipe de trabalho (Farrahi *et al.*, 2000). Neste sentido, Miyazaki (1998) apresenta a seguinte notação para classificar os tipos de dispositivos lógicos reconfiguráveis:

a) *Dispositivos de lógica configurável*: podem ser customizados uma única vez.
b) *Dispositivos de lógica reconfigurável*: podem ser customizados mais de uma vez. Eles adotam tecnologias EPROM, EEPROM ou FLASH e podem ser reprogramados uma vez montados em uma placa de circuito impresso.
c) *Dispositivos de lógica dinamicamente reconfigurável*: permitem a programação durante a operação, mesmo após a montagem em uma placa de circuito impresso. Essa propriedade é chamada de reconfiguração *in-board*.
d) *Dispositivos de interconexão dinamicamente reconfigurável*: é o termo para dispositivos interconectados que podem ser programados por conexões pino a pino após a montagem em placa de circuito impresso.
e) *Dispositivos de lógica virtual*: apresentam capacidade parcial de reconfiguração. Apenas uma parte do dispositivo pode ser reprogramada enquanto a outra executa alguma função definida. Em outras palavras, diferentes circuitos lógicos podem partilhar ao longo do tempo a mesma parte do dispositivo. São também chamados de dispositivos de lógica adaptativa ou dispositivos de lógica compartilhada.

Entre as principais vantagens da computação reconfigurável estão o paralelismo, a velocidade, a capacidade de atualização, a padronização de plataformas, assim como a amortização do custo de desenvolvimento e alto desempenho.

2.7 CONCLUSÕES DO CAPÍTULO

Neste capítulo de fundamentação teórica foram apresentadas, por meio da revisão da literatura, as partes fundamentais que compõem uma plataforma para desenvolvimento de sistemas embarcados, abrangendo capacidade de reconfiguração. Foram apresentados os princípios do controle e da automação e como eles têm ganhado espaço no dia-a-dia da sociedade da informação, hoje, tão dependente de dispositivos eletrônicos.

Em seguida, foram apresentados os sistemas embarcados os quais são definidos como sistemas computacionais para uso específico ou dedicados e possibilitaram a rápida abrangência e a massificação de sistemas eletrônicos à vida das pessoas. Sistemas embarcados já fazem parte do cotidiano das pessoas embutidos nos mais diversos dispositivos. Há certo tempo eles vêm ganhando força especialmente devido à economia de energia, portabilidade e baixo custo. Os seguimentos mais relevantes são telefonia móvel, automobilística, domótica e mais recentemente, a computação ubíqua.

Alguns autores afirmam até que computadores pessoais e estações de trabalho se tornarão obsoletos, pois o acesso computacional à informações estará disponível em todos os lugares, nos mais diversos dispositivos (Weiser, 1993).

Esse fenômeno observado pela integração sutil e natural entre o homem e os dispositivos que agregam as tecnologias de informações e comunicações dá-se o nome de ubiquidade. São os denominados sistemas ubíquos que definem a computação portátil no cotidiano das pessoas.

Essa interação homem-máquina não objetiva ser imperceptível, mas sim natural, de uma forma que as pessoas percebam que estão interagindo com um dispositivo, mas tão naturalmente quanto uma conversa com um amigo.

Essa evolução observada, especialmente nas ultimas décadas, só foi possível pela miniaturização dos componentes eletrônicos associada ao elevado poder computacional e baixo custo. Assim, deu-se destaque ao principal componente dos sistemas embarcados, os processadores ou microcontroladores, especialmente aqueles utilizados na elaboração da plataforma, *ARM7,* bem como tecnologias reconfiguráveis como os *FPGAs*.

O *ARM*, considerado um processador de propósitos gerais – *GPP*, segue a arquitetura de von Neumann e apresenta-se como um processador RISC de 32 bits, *pipeline* de três estágios e arquitetura de registradores *load*/*store*. Esses *CIs* já se consagraram no mercado por seu grande potencial de aplicações

devido à alta performance associada ao baixo custo. Alguns aparelhos que atualmente utilizam o *ARM7* são: *Apple iPod, Nintendo Ds e Game Boy Advance*, vários celulares Nokia, Lego *Mindstorms* NXT, SEGA *Dreamcast*.

Já o *FPGA* representa a nova geração de soluções para ambientes embarcados. Sistemas de desenvolvimento baseados em computação reconfigurável (sistemas de *hardware* reconfigurável) apresentam características baixo consumo de energia, paralelismo de operações, capacidade de integração, flexibilidade e operação modular.

Por fim, destacou-se a importância e vantagens da utilização de computação reconfigurável em comparação com tecnologias baseadas unicamente nas maquinas de von Neumman, especialmente em consumo energético. Por outro lado, os *GPP*s contam com a vantagem de apresentarem melhores resultados computacionais em relação a cada dólar investido (Hamada *et al.*, 2009). A união das duas tecnologias em uma única plataforma dá a possibilidade ao usuário de extrair o melhor de cada uma delas.

Tomando como referência essa revisão bibliográfica, partiu-se para a elaboração da metodologia para a fabricação da plataforma. No Capitulo 3, serão apresentados os esquemas conceituais de concepção do projeto, passando pela metodologia adotada, seleção dos componentes, fabricação de placas, montagem da plataforma e finalizando com os testes.

CAPÍTULO 3
PROPOSTA DE METODOLOGIA PARA A ELABORAÇÃO E TESTES DA PCI

A metodologia utilizada para a elaboração da plataforma foi elaborada por meio da experiência de desenvolvimento por parte dos membros da equipe em outros projetos relacionados à confecção de plataformas PCI.

Em primeiro lugar deve-se fazer um levantamento de requisitos, em seguida elaborar um esquemático contendo as partes ou funcionalidades essenciais ao projeto. Em seguida, desenvolver todos os circuitos constituintes do sistema para se obter a plataforma desejada, que atenda aos requisitos predefinidos.

Em conjunto com o desenvolvimento do sistema, deve ser feita a seleção dos fornecedores dos componentes a serem utilizados. Finalizada a engenharia do projeto, segue-se com a elaboração da topologia da placa de circuito impresso – PCI[11].

Feito o *layout* da placa, segue-se com a fabricação dela e a montagem dos componentes por meio da contratação de empresas especializadas. Por fim, são feitos os testes de funcionalidade do sistema e observados os resultados obtidos.

3.1 LEVANTAMENTO DE REQUISITOS

Segundo as melhores práticas aplicadas ao gerenciamento de projetos, um dos primeiros passos a ser dado para a elaboração de um sistema, seja ele qual for, é o levantamento de requisitos do sistema (Vargas, 2009). Antes disso, pode ser feito um estudo de viabilidade técnico-financeiro em separado ou opcionalmente este estudo pode ser incorporado à fase de levantamento de requisitos.

O termo, levantamento de requisitos, refere-se ao um processo que engloba todas as atividades que contribuem para a produção de um documento de requisitos e sua manutenção ao longo do tempo. Com isso, busca-se levantar

[11] Em inglês, *PCB* - *Printed Circuit Board*

todas as características essenciais e não essenciais que podem ou não serem incorporadas ao projeto.

Por meio do levantamento de requisitos busca-se obter as mais diversas informações referentes ao projeto, como, por exemplo, requisitos do utilizador, características dos usuários, requisitos do sistema, design da aplicação, levantamento de possíveis dificuldades, usabilidade e amigabilidade da solução, performance, requisitos operacionais, segurança, restrições culturais, legais, políticas, etc.

É interessante que sejam separados os requisitos essenciais e não essenciais ao projeto. Assim, estabelece-se que nenhum requisitos essencial poderá ser negligenciado durante a execução. Por outro lado, buscar-se-á atender na medida do possível (viabilidade técnica, financeira e operacional) os objetivos não essenciais. Essa distinção facilitará uma eventual fase de aceitação do projeto por parte do demandante.

Os requisitos definidos na fase inicial do projeto servirão como referencia para a elaboração dos objetivos e metas que deverão ser seguidos por toda a equipe de desenvolvimento. Nota-se a importância desta etapa e a necessidade que ela seja suficiente para dirimir qualquer dúvida durante o desenvolvimento do sistema. Uma vez estabelecidas, qualquer alteração posterior poderá impactar, de maneira não linear, o cronograma e recursos de todo o projeto. Por isso, é extremamente importante que esses requisitos sejam devidamente documentados e assinados por todos aqueles envolvidos na demanda e execução do projeto.

A partir desses requisitos parte-se para a elaboração do projeto conceitual.

3.2 PROJETO CONCEITUAL DO SISTEMA

Em seguida deve-se elaborar um esquema conceitual com todos os módulos essenciais que constituem a solução proposta. Nesse conceito estarão os componentes básicos para uma abordagem adequada de desenvolvimento do sistema.

O projeto conceitual é uma representação gráfica, simplificada, do que seria a solução proposta. Recomenda-se a utilização de diagrama de blocos ou fluxograma para indicar as principais partes do projeto.

Este projeto é importante porque possibilita visualizar de maneira direta as principais partes constituintes do sistema ou solução a ser implementada. Este conceito deve ser discutido entre os desenvolvedores e a parte demandante,

de modo a ficar clarividente para todos os envolvidos na solução, os limites do projeto.

3.3 ELABORAÇÃO DOS CIRCUITOS ELETRÔNICOS

A elaboração dos circuitos eletrônicos é a fase que se segue à definição do projeto conceitual. A execução dessa tradução dos módulos representativos em sistemas elétricos e eletrônicos deve ser feita com base na teoria de eletrônica, microeletrônica, documentação dos componentes, notas de aplicação dos fabricantes e experiência da equipe técnica.

A elaboração desses circuitos pode ser bastante facilitada pela utilização de ferramentas CAD/EDA que servem para auxiliar os projetistas de circuitos. Como exemplo, pode-se citar o *software Altium Designer*, que conta com ferramentas de auxilia à elaboração de circuitos esquemáticos (elétricos e eletrônicos), projeto da placa de circuito impresso e ainda com a representação 3D da solução.

Esta ferramenta possibilita que o projetista tenha à sua disposição todos os meios necessários para explorar as mais diversas técnicas e tecnologias de projeto de sistemas eletrônicos. Além de poder fazer uso de uma imensa e vasta biblioteca de componentes e dispositivos, que já são disponibilizados no pacote de ferramentas do *software*. É possível ainda criar novos componentes, assim como novas bibliotecas próprias as quais podem ser vinculadas a um projeto específico.

3.4 SELEÇÃO E COMPRA DE COMPONENTES DO SISTEMA

Todos os componentes devem ser selecionados durante a fase de projeto levando-se em conta os projetos de circuitos elaborados e qualidade exigida em cada projeto.

Uma vez definido o valor ou características dos componentes a serem utilizados, deve-se fazer uma pesquisa junto a fornecedores nacionais e internacionais, segundo requisitos de disponibilidade de fornecimento, facilidade de aquisição, confiabilidade, quantidade em estoque, continuidade de fabricação, valor em relação aos similares e compromisso ambiental de modo a realizar a melhor escolha. O mercado de componentes é muito dinâmico e isso

se reflete diretamente na diversidade e complexidade dos componentes disponíveis atualmente.

No contexto da dinâmica do mercado de componentes eletrônicos, à medida que um componente começa a perder mercado, seja por desempenho, concorrência, preço ou qualquer outro motivo, ele é rapidamente substituído por outro melhor. Quando a indústria não consegue melhorá-lo a ponto de concorrer no mercado, ele é simplesmente descontinuado. Esse é um grande problema para os desenvolvedores de *hardware*, pois precisam selecionar um componente aprovado pelo mercado e com expectativa de continuidade de produção nos próximos anos. Normalmente, antes de ser descontinuado ou substituído, o fabricante alerta aos fornecedores da mudança que, mediante consulta, informa aos seus clientes. Além disso, precisa-se contar com um preço justo, assim como atingir um desempenho adequado à aplicação desejada.

Combinar esses fatores é uma tarefa complexa, mas com certa experiência e trabalho de pesquisa é possível chegar a resultados satisfatórios, que cumpram algum tipo de relação positiva de custo/benefício. Um dos principais pontos a se observar é se, caso o componente selecionado for descontinuado, será possível prosseguir com o projeto utilizando outros meios, como substitutos ou concorrentes, sem grandes mudanças estruturais. Dessa forma, busca-se garantir a continuidade do desenvolvimento pelos próximos anos.

Observando esses pontos, garante-se maior facilidade para a manutenção da plataforma, confecção de novas unidades ou mesmo as melhorias a partir do projeto base.

Outro ponto relevante a ser considerado é que muitas vezes o valor estabelecido para o circuito não é um valor comercial encontrado, então identificar isso durante a elaboração de esquemáticos facilita propagar as mudanças decorrentes dessa alteração para o restante do sistema. E ainda, buscar escolher componentes padrão, ou seja, evitar escolhas muito específicas que possam inviabilizar atualizações ou reproduções do sistema no futuro.

3.5 PROJETO DA PLACA DE CIRCUITO IMPRESSO

A placa de circuito impresso é o meio mais comum e prático usado para montagens definitivas de sistemas eletrônicos. Os processos de confecção de placas de circuito impressos são variados, podendo distinguir-se basicamente

em processos caseiros e processos profissionais mais elaborados, os quais possibilitam uma produção industrial em larga escala (Pedroso, [S.d.]).

O termo circuito impresso é derivado do método original, onde um padrão gravado ou estampado é usado para mascarar as áreas desejadas, de uma placa de cobre. O objetivo principal da placa de circuito impresso é a circulação da corrente de um circuito por meio de uma fina camada de cobre ou outro condutor, fixando os componentes, melhorando sua distribuição e diminuindo o espaço necessário à montagem (Pedroso, [S.d.]).

A camada condutora está fixada sobre uma base isolante, por exemplo, fenolite ou fibra de vidro, sendo que na maior parte das aplicações industriais utilizam-se duas ou mais camadas de cobre, separadas pela base isolante (Pedroso, [S.d.]).

O processo de fabricação da placa de circuito impresso pode ser dividido em duas etapas distintas e igualmente importantes: (a) elaboração do *layout*, (b) impressão da placa.

Elaborar o *layout* de uma placa de circuito impresso não é um processo muito simples. Por exemplo, na ferramenta de auxilio ao desenvolvimento CAD/EDA *Altium Designer,* ao final do projeto esquemático dos circuitos é possível exportar este projeto para a interface de elaboração de *layout*. Entretanto, um projeto bem sucedido requer o cumprimento de muitos requisitos e restrições, os quais só podem ser atingidos por um projetista experiente, sobretudo quando se trata de um projeto de múltiplas camadas.

A ferramenta disponibiliza total integração e compatibilidade entre os projetos e, de maneira bem simples, é possível criar o *layout* da placa. A primeira coisa a se fazer é o posicionamento dos componentes. Mas aí se encontra uma área crítica para o projeto como um todo. A moderna tecnologia dos circuitos integrados tem imposto requisitos cada vez mais rígidos ao projeto térmico e magnético de sistemas eletrônicos. Este fenômeno é evidenciado pelo elevado aumento de densidade dos CIs, potências dissipadas, frequência de operação, além da constante redução de seus empacotamentos (*package*).

Antagonicamente, com o aumento da frequência de operação dos componentes é desejável que eles estejam posicionados o mais próximo possível uns dos outros de forma a manter a integridade do sinal elétrico, e minimizar a perda de informações nas ligações do sistema. Por outro lado para aumentar o fator de dissipação de potência é necessário mantê-los distantes uns dos outros para evitar superaquecimento (Pereira Jr; Sousa; Vlassov, [S.d.]), e ainda restringir possibilidades de interferências eletromagnéticas. Com todos esses problemas, obter um posicionamento ideal é uma tarefa

complexa, pois os algoritmos de projeto devem lidar com funções de otimização de múltiplas variáveis.

Como a confiabilidade de um componente depende, entre outros fatores, da temperatura de operação, é importante que o projeto PCB busque garantir que a máxima temperatura de seus componentes seja a menor possível (Xu; Li; Jiang, 2009). Apesar de todas as nuanças existentes e ainda possuir *softwares* voltados para a tarefa de posicionamento de componentes (Guan; Guo, 2010; Xu; Li; Jiang, 2009), ela é carregada de decisões resultantes da experiência pessoal do projetista (Pereira Jr; Sousa; Vlassov, [S.d.]).

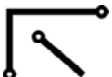
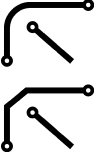
Em adição, assim como existem *softwares* para posicionamento de componentes, existem também outros direcionados para o posicionamento de furos na placa (Guan; Guo, 2010). Na prática, qualquer fator que possa gerar algum tipo de interferência nos componentes ou nos sinais que percorrem a placa deve ser tratado com especial atenção.

Outro fator a ser considerado, talvez o mais crítico, são as trilhas condutoras de sinais pois elas podem se comportar como antenas, amplificando e capturando sinais. A começar pela definição de espessura e largura. Para se encontrar a espessura ideal das trilhas existem inúmeras maneiras, uma delas é definida pelas equações 1 e 2 (Diodes Incorporated, 2010):

Trilhas internas : $I = 0.024 \times dT^{0.44} \times A^{0.725}$ (1)

Trilhas externas: $I = 0.048 \times dT^{0.44} \times A^{0.725}$ (2)

onde

I = Corrente máxima, em Amperes;
dT = Gradiente de temperatura acima da ambiente, em °C;
A = Área de sessão transversal, em mils².

As ferramentas CAD/EDA atuais fazem esses cálculos de maneira mais simplificada mediante a inserção de regras e restrições referente à corrente, temperatura, áreas de isolamento, etc. Além da definição da área de sessão transversal da trilha é preciso traçar um caminho adequado para fazer a ligação entre os componentes, posicionando-a corretamente de modo a evitar interferências e ruídos. Este processo é denominado, na literatura, de roteamento (Naveda; Chang; Du, 1986). As soluções mais completas de *EDA's* oferecem ferramentas de auto-roteamento (*autoroute*), mas mesmo após utilizadas é recomendável revisar as ligações para eliminar eventuais anomalias (Ajay Kumar, 1995). Novamente, a experiência do profissional que trabalha como *layoutista* de projeto é muito relevante. A Tabela 3 apresenta, a

título de exemplo, algumas restrições de projeto ao se posicionar as trilhas condutoras (Pedroso, [S.d.]).

Tabela 3 – Considerações e justificativas para soluções de *layout* de PCI.

Problema	***Errado***	***Correto***	***Justificativa***
Filetes largos ou grandes superfícies			Melhor condição de soldagem reduz o espalhamento de solda.
Ligações em triângulo			Melhor distribuição de corrente aumento da área útil e redução do número de pontos de chegada nas ilhas propiciando melhor soldagem.
Cruzamentos			Redução do número de pontos de chegada nas ilhas (de 3 para 1) propiciando melhor soldagem.
Curvas			Redução da indutância das trilhas melhor estética.
Cruzamento sem ilhas			Melhor distribuição de corrente e melhor estética.
Correspondência entre trilhas e ilhas			A dimensão das ilhas deve corresponder à largura das trilhas (o diâmetro deverá ser aproximadamente o dobro da largura da trilha).

Fonte: http://www.nabucoeletronica.com.br/files/pci.pdf

Outro fator a ser considerado é o material de fabricação da placa. Existem inúmeras composições no mercado podendo, inclusive serem combinadas, tendo cada uma, propriedades específicas para cada necessidade. A exemplo, pode-se citar (Micropress, [S.d.]):

a) CEM-1: Resina Epóxi, Fibra de vidro (na superfície) e Papel (no interior). Chamado de *Composite*;

b) CEM-3: Resina Epóxi e Fibra de vidro não trançada;

c) FR-1: Resina fenólica e Papel;

d) FR-2: Resina fenólica e Papel;

e) FR-4: Resina epóxi e Tecido de fibra de vidro;
f) RO 3003: PTFE e Cerâmica;
g) RO 4003: Fibra de vidro com cerâmica e hidrocarboneto;
h) RT / Duroid 5880: PTFE e Microfibra de vidro;
i) RT / Duroid 6010: PTFE e Cerâmica;
j) TMM: Cerâmica e hidrocarboneto;
k) Ultralam 2000: PTFE e Fibra de vidro trançada.

A escolha deste fator depende de variáveis como utilização, *design*, variação de temperatura, resistividade, impedância, faixa de frequência e custo (Micropress, [S.d.]). Normalmente, essa escolha é feita a partir de uma consulta ao fabricante de placas que, de acordo com as variáreis apresentadas, sugere a melhor opção.

Recomenda-se a terceirização desta etapa por meio da contratação de empresas ou profissionais especializados na execução deste tipo de serviço.

3.6 MANUFATORA DA PLACA DE CIRCUITO IMPRESSO

O custo associado a um projeto com falhas é muito alto. Enquanto ainda não fabricada a PCI, o custo do projeto é basicamente devido ao tempo investido dos projetistas e mudanças ainda são facilmente incorporadas. A partir do momento que se define o projeto de fabricação da placa incorre-se, por consequência, ao custo da fabricação, compra de componentes conforme desenhos definidos, montagem de componentes e homem-hora para testes, etc.

Dessa forma, tudo o que puder ser feito de forma a minimizar a possibilidade de falhas no projeto precisa ser feito antes de finalizado o *layout* de fabricação da placa. A partir daí, mudanças, quando possíveis, são pontuais pois todo o resto estará em função do *hardware* que a partir deste momento torna-se invariável.

Recomenda-se a terceirização desta etapa por meio da contratação de empresas nacionais especializadas.

3.7 MONTAGEM DE COMPONENTES DO SISTEMA

A última etapa para a obtenção da plataforma é a montagem dos componentes na PCI. Esta etapa também requer mão-de-obra e ferramental especializados. Os componentes utilizados são, em sua maioria, *SMD* (*surface mount device*) que, se por um lado são mais baratos e reduzem a densidade da placa, por outro exigem ferramental próprio para manuseio e soldagem.

A utilização de técnicas ou procedimentos inadequados durante o processo de soldagem dos componentes pode representar risco para todo o sistema. O próprio manuseio dos circuitos impressos, se não realizados de forma adequada, é suficiente para danificá-los. Dispositivos danificados podem prejudicar outros e assim danificar todo o sistema.

Recomenda-se a terceirização desta etapa por meio da contratação de empresas nacionais especializadas.

3.8 METODOLOGIA DE TESTE DO SISTEMA PROJETADO

O primeiro teste a ser realizado é no *hardware* da PCI pelo próprio fabricante da placa de circuito impresso, antes da soldagem dos componentes. Esse teste consiste na verificação de boa condutibilidade das trilhas, áreas de isolamento (*clearance*), assim como a verificação da ausência de curtos-circuitos. Recomenda-se, ainda assim, uma checagem manual, especialmente das ligações referentes à alimentação de cada componente para verificar se todas as trilhas estão interligadas conforme o projeto.

A metodologia adotada para testes do sistema deve seguir a lógica de desenvolvimento dele. Quando são utilizados sistemas modulares, desacoplados por meio de *jumpers*, os testes de funcionamento se tornaram bastante descomplicados, tendo em conta a possibilidade de isolar partes do sistema pela remoção de *jumpers* na placa. Dessa forma deve-se realizar os testes em cada módulo, isoladamente (quando possível) de forma a garantir seu correto funcionamento antes de entrar na fase dos *testes de integração*.

Após a montagem dos componentes, parte-se para o teste de funcionamento do sistema propriamente dito (testes modulares e de integração). O isolamento da fonte de alimentação para que ela seja testada antes de qualquer outro módulo é importantes pois caso haja alguma falha de projeto, ou mesmo defeito de componente, uma eventual sobrecarga não é transmitida para o restante do sistema.

Constatado o correto fornecimento de tensão pelos módulos de alimentação, segue-se para os testes dos demais módulos de todo o sistema. Quando não for possível o teste em determinado módulo isoladamente, ele deve ser testado após constatado o correto funcionamento do módulo principal, ou seja, o que contém o núcleo do sistema, ao qual este aquele módulo está vinculado.

O módulo principal, ao ser alimentado, inicialmente deve-se verificar se todos os pinos de alimentação estão com tensão condizente com o projeto. Feita essa verificação passa-se para a fase de programação.

Antes de iniciada a programação, deve-se verificar se a arquitetura utilizada disponibiliza ferramentas de desenvolvimentos e programação, muitas vezes oferecidas pelos fabricantes dos dispositivos. A configuração inicial de ferramentas desse tipo pode ser um tanto trabalhosa mas os tutorias dos fabricantes ou disponibilizados na internet facilitam esse trabalho.

Testes simples como ligar e desligar *LED's por meio dos botões*, é possível observar o funcionamento tanto lógico quanto das estruturas físicas (botões, LED's, RS232) relacionadas neste processo.

A seguir, os mesmos procedimentos devem ser repetidos, para outros módulos ou demais núcleos que constituem a solução. Os resultados visuais e a leitura de sinais elétricos indicam o bom funcionamento dos núcleos e parte do *hardware* de apoio a ele.

Por último, devem ser feitos testes de comunicação utilizando todos os subsistemas, ou seja, verificar o funcionamento integrado da plataforma. Algumas trilhas devem ser selecionadas para os testes, às quais poderão comprovar a efetiva troca de informações. Essa verificação pode ser feita por meio do acompanhamento de sinais de tensão no osciloscópio e também de maneira visual.

Seguindo esses procedimentos, as partes principais constituintes do sistema poderão ser testadas e verificado seu funcionamento devido. A aplicação desta metodologia no desenvolvimento da plataforma e dos testes realizados podem ser acompanhados no CAPÍTULO 4 e no CAPÍTULO 5.

3.9 RESUMO DA METODOLOGIA PROPOSTA

Para facilitar a aplicação da metodologia ora proposta, fez-se um resumo dos procedimentos básicos necessários à elaboração de um sistema embarcado, desde a fase de concepção até os testes. Segue:

A. Levantamento de requisitos do sistema: fazer uma seleção de todos os requisitos necessários e desejados do sistema. Buscar ser o mais detalhado possível pois facilita a escolha de componentes e a implementação ou não de subsistemas ou módulos. Uma vez estabelecidos, criar um termo de aceitação formal junto ao cliente ou demandante (quando for o caso) e deixar claro que qualquer alteração posterior nesses requisitos irá impactar, de forma não linear, os prazos, custos e recursos do projeto.
B. Elaboração do projeto conceito do sistema contendo os módulos que o constituirão: o projeto conceitual é uma representação gráfica, simplificada, do que seria a solução proposta. Este projeto é importante porque possibilita visualizar de maneira direta as principais partes constituintes do sistema ou solução a ser implementada. Este conceito deve ser discutido entre os desenvolvedores e a parte demandante, de modo a ficar clarividente para todos os envolvidos na solução, os limites do projeto.
C. Com base no conjunto de requisitos do sistema, definir os componentes principais da solução (núcleos): esses componentes devem ser selecionados de modo a atender todos os requisitos necessários e, na medida do possível, os desejados.
D. A partir da definição dos núcleos, elaborar os módulos estritamente necessários ao funcionamento deles: buscar criar cada módulo de maneira independente e, quando possível, isolando-o por meio de *jumper*. A modularização do sistema facilita os processos de testes, correção de erros, compreensão, atualização do sistema tanto para quem o desenvolveu quanto para terceiros.
E. Elaborar todos os módulos constantes na tabela de requisitos necessários e, quando viável, os desejados.
F. Compatibilizar os módulos elaborados com seu respectivo núcleo: compatibilizar diz respeito a uniformizar, por exemplo, tensões e corrente em cada módulo. Configurar corretamente entradas e saídas de sinais em cada módulo de modo a evitar eventuais sobrecargas ou curto circuitos.
G. Escolha de componentes: durante a elaboração dos circuitos eletrônicos é aconselhável já fazer a escolha dos componentes junto a fornecedores. Muitas vezes o valor estabelecido para o circuito não é um valor comercial e identificar isso durante a elaboração facilita propagar as mudanças decorrentes dessa alteração para o restante do sistema. Escolher componentes padrão, ou seja, evitar escolhas muito específicas que possam inviabilizar atualizações ou reproduções do sistema. Escolher componentes capazes de suprir as necessidades do sistema com margem de segurança. Verificar a disponibilidade e quantidade em estoque dos componentes para evitar atrasos na entrega. Verificar a existência de substitutos. Compatibilizar a escolha de todos os componentes.
H. Escolha dos fornecedores: existem vários distribuidores de componentes eletrônicos no cenário nacional e internacional (vide sessão 3.4), cabe ao projetista escolher aquele que melhor pode lhe atender. Recomenda-se fazer uma cotação com todos e verificar as vantagens e

desvantagens de cada um. Há 6 anos o autor deste projeto trabalha com a digikey.com, apesar de cobrar um alto custo para envio (cerca de U$60), garante a entrega em menos de uma semana (ver disponibilidade no estoque) e contem a maior variedade de componentes.

I. Elaboração do *layout* da PCI – posicionamento dos componentes: somente depois de finalizados todos os projetos dos circuitos é que parte-se para o posicionamento de componentes. As etapas anteriores precisam ser finalizadas, pois o acréscimo ou retirada de algum componente influi no *layout* da placa, posicionamento de trilhas bem como suas dimensões.
J. Elaboração do *layout* da PCI – dimensionamento e posicionamento de trilhas, furos e vias: uma vez posicionados os componentes faz-se o dimensionamento das trilhas, furos e vias, bem como seu posicionamento. O dimensionamento pode ser criado por meio de definições de regras e restrições a que as trilhas, furos e vias serão submetidas durante o processo de roteamento. Essas regras são criadas a partir da definição de corrente máxima, frequência dos sinais, interferências externas e variações de temperatura. Criadas as restrições pode-se solicitar ao *software* CAD/EDA que faça o auto-roteamento, que seria o processo automático para o posicionamento de trilhas, furos e vias. Entretanto, todo o processo de *layout* da PCI, além de ser bastante trabalhoso, ainda exige um profissional com experiência para a realização de projetos confiáveis e imunes a interferências. Dependendo da complexidade do projeto, mais de uma camada pode ser necessária para comportar todos os sinais, o que impactará significativamente no valor da fabricação da PCI. Recomenda-se que esse trabalho seja realizado por profissionais com larga experiência nesse tipo de atuação.
K. Compra de componentes. Finalizado o *layout*, faz-se a compra dos componentes. Pode ser que haja um lapso temporal entre a elaboração dos circuitos, e escolha dos componentes, e a efetiva compra deles e nesse período pode ser algum componente tenha acabado no fornecedor, tenha sido descontinuado, etc. Então antes de mandar fabricar a PCI, é necessário checar se é possível comprar a lista de componentes anteriormente selecionada. Caso algum imprevisto tenha ocorrido, ainda é possível atualizar, pontualmente, o projeto da PCI de maneira relativamente simples.
L. Fabricação da PCI. Confirmada a compra dos componentes pode-se fazer a fabricação da PCI. Conforme descrito na sessão 4.4 existem vários fabricantes nacionais de PCI, cabe ao projetista realizar a cotação e comparar as vantagens e desvantagens de cada um deles. Caso sejam utilizadas mais de duas camadas para a transmissão de sinais, a quantidade de fabricantes nacionais será significativamente reduzida e o valor substancialmente elevado. Recomenda-se solicitar que o fabricante realize testes de funcionamento das placas para verificar a existência de alguma ligação inadequada. Normalmente este serviço é cobrado à parte.

M. Montagem dos componentes na PCI. Diversas são as empresas que prestam serviço de montagem de componentes, uma busca rápida na internet é suficiente para localizá-las. É necessário apenas que sejam enviados os componentes e as placas nas quais serão feitas a montagem. Recomenda-se a contratação de profissionais com larga experiência nessa atividade de modo a reduzir a possibilidade de danificar tanto componentes como placas por meio de soldas mal feitas, manuseio incorreto ou procedimentos inadequados adotados.
N. Realização de testes. Nesse ponto tem-se o *hardware* todo montado e pronto para os primeiros testes. Caso tenha optado por modularizar o projeto e isolar os blocos por meio de *jumpers* essa fase fica significativamente mais segura de ser feita. Inicia-se realizando testes de conexões para verificar se todas as ligações estão corretas. Em seguida alimenta-se o sistema e testa-se o módulo de alimentação. Só então cada núcleo poderá ser alimentado com a fonte já testada e os primeiros testes de programação poderão ser realizados. Em seguida testa-se a integração entre os módulos e núcleos de modo a verificar o funcionamento do sistema com um todo.
O. Verificação de resultados. Por meio dos testes pode-se verificar o funcionamento do sistema e checar se todos os requisitos inicialmente estabelecidos foram atingidos.
P. Temo de encerramento de projeto. Finalizada a fase de testes, é necessário que o projeto seja detalhadamente documentado de forma a viabilizar eventuais correções ou aperfeiçoamentos. É muito importante que essa documentação seja feita de forma continua e objetiva, ao final de cada processo, desde a fase inicial até chegar aos últimos testes de funcionamento. Essa documentação irá compor o termo de encerramento de projeto em que o cliente ou demandante deve assinar e dar como cumpridas as obrigações inicialmente estabelecidas.

3.10 CONCLUSÕES DO CAPÍTULO

Neste capítulo foram apresentadas as metodologias utilizadas para a elaboração do projeto, seleção dos componentes, fabricação de placas, montagem e testes da plataforma.

Constatou-se que diversos fatores devem ser observados em cada etapa de desenvolvimento para se conseguir resultados satisfatórios e a garantia de possível continuidade do projeto.

Durante a elaboração do esquema conceitual do sistema, destaca-se a constante preocupação por utilizar subsistemas modulares. Em primeiro lugar, a modularidade do projeto facilita o estudo e compreensão dele por terceiros. Além disso, possibilita futuros desenvolvedores utilizarem os subsistemas em outros projetos ou mesmo aperfeiçoar algum módulo da plataforma ora apresentada com certa facilidade.

Para a elaboração dos circuitos eletrônicos foram selecionadas soluções disponíveis no mercado de auxílio a projetos eletrônicos, são ferramentas CAD/EDA. No caso em tela, foi utilizado o *Altium Designer, versão* 6.9.

Na seleção de componentes, ressalta-se a preocupação na escolha de componentes fundamentada na tradição dos fabricantes, disponibilidade, facilidade de aquisição, confiabilidade, volume em estoque, custo total e compromisso ambiental. A seleção adequada de componentes garante maior facilidade para a manutenção da plataforma, confecção de novas unidades ou mesmo as melhorias a partir do projeto base.

No processo de fabricação das placas de circuito impresso foram vistas diversas questões que influenciam fortemente o funcionamento e desempenho do projeto, independente da exatidão dos circuitos eletrônicos. São fatores como posicionamento de componentes, vias, espessura de trilhas condutoras, posicionamento e *layout* das trilhas além da composição da placa de circuito impresso. Foi visto também que apesar das ferramentas CAD/EDA disponibilizarem mecanismos automatizados de projeto, a experiência do projetista é fundamental no momento revisional.

Por fim, a definição de uma metodologia para a fase de testes é importante para que eventuais falhas nos subsistemas, ou módulos de projeto, possam ser identificados e corrigidos sem colocar em risco o restante da plataforma. O teste de cada subsistema possibilita verificar o adequado funcionamento de cada bloco e após a integração, fazer a mesma verificação para o conjunto da plataforma.

Todo o processo metodológico elaborado para o desenvolvimento do sistema serviu de guia para a execução de todas as etapas subsequentes. Definida a metodologia, partiu-se para o desenvolvimento prático do projeto, conforme descrito no CAPÍTULO 4.

CAPÍTULO 4
DESENVOLVIMENTO DO PROJETO

Com o objetivo de se criar uma plataforma reconfigurável para dar suporte às possíveis aplicações descritas no CAPÍTULO 2, partiu-se para a seleção adequada de dispositivos essenciais ao projeto. De fato, o produto resultante desse sistema também é uma plataforma para desenvolvimento de futuros sistemas no âmbito dos sistemas embarcados, que não se sabe a *priori* que tipo de processamento requisitarão. Sinais de áudio e vídeo, por exemplo, podem precisar de elevada capacidade de processamento, ao passo que o monitoramento de um ambiente residencial não exigirá tanto.

O desafio inicial foi adequar os componentes que pudessem atender uma grande parte de projetos de processamento de dados/sinais e, na medida do possível, reduzir ao máximo os custos relacionados às aplicações. Para isso, foram tomados como referência alguns projetos que vinham sendo desenvolvidos entre os alunos de engenharia da Universidade de Brasília. Também foram considerados alguns *kits* de desenvolvimento disponíveis no mercado que pudessem atender à maioria das demandas do curso de Engenharia Mecatrônica.

A partir disso, optou-se por desenvolver um sistema composto por dois núcleos de processamento. O primeiro composto por um processador *ARM7* para servir como núcleo principal de gerenciamento do sistema. O segundo, integrando uma unidade reconfigurável, *FPGA*, que pode ser dedicado ao processamento de sinais, uma vez que isso requer maior capacidade computacional.

A seleção de qual *ARM7* e qual *FPGA* utilizar acompanhou as características técnicas dessas unidades de processamento e, em parte, algumas ferramentas disponíveis em laboratório para testes.

Pelo aspecto técnico, o *ARM7* se mostrou bastante satisfatório ao fim desejado. Sem dúvida, ele é uma boa escolha para as finalidades de controle devido às opções de interface, performance e preço que é equivalente a alguns microcontroladores de 8-bits. Como o *ARM7* não é produzido por apenas um fabricante, então optou-se por utilizar os *CIs* do fabricante NXP pelo fato de

haver, no laboratório, alguns *kits* de desenvolvimento utilizando um *ARM7* da família LPC2000 desse fabricante, estando disponível toda a documentação de apoio. Além disso, o valor dele é equivalente ou, em alguns casos, inferior aos dos concorrentes.

Definido o *ARM*, a segunda etapa foi a seleção do *FPGA*. Seguindo o mesmo raciocínio da seleção do ARM, optou-se por escolher um dispositivo que atendesse às demandas do projeto e, de preferência, que fosse utilizado nos laboratórios da UnB para poder aproveitar o conhecimento e experiência acumulados ao longo dos anos. Como resultado, optou-se por utilizar o XC3S500E do fabricante Xilinx, com 500.000 *gates*. Este *FPGA* atende, satisfatoriamente, à grande parte dos projetos de controle e automação de pequeno e médio porte, tendo em conta a experiência obtida durante vários anos no Laboratório de Eletrônica do Graco.

A partir da seleção dos componentes de controle e processamento, partiu-se para a elaboração dos periféricos, ou seja, os circuitos que dão suporte aos componentes principais. Prezando pela modularidade do projeto, optou-se por desenvolver cada periférico separadamente. A modularidade facilita o processo de teste do projeto, a sua manutenção, revisão e evolução de todo o sistema, além de torna-lo mais compreensível por terceiros.

Cada módulo foi elaborado tendo como referência o controlador a que ele irá servir. Cada controlador (*ARM7* e *FPGA*) possui os respectivos circuitos de referência fornecidos pelo fabricante a partir da documentação de apoio à elaboração de projetos. Por outro lado, todos os subsistemas e respectivos componentes foram cuidadosamente selecionados a partir das sugestões dos fabricantes para que houvesse compatibilidade no conjunto. Por exemplo, cada controlador requer uma alimentação com características específicas e, para isso, foram implementadas duas fontes independentes que atenderão, prioritariamente, a seu respectivo controlador.

Os módulos que dão suporte ao *ARM* são: *(a) ARM - Alimentação, (b) ARM - Memória, (c) ARM - Interfaces e (d) ARM - Comunicação*. Já o *FPGA* é apoiado pelos seguintes módulos: *(a) FPGA - Alimentação, (b) FPGA – Memória, (c) FPGA - Interfaces e (d) FPGA – Comunicação*. Além desses, os controladores mantêm canais para comunicação entre si, para troca de informações e dados, isso faz com que cada núcleo seja visto também como um módulo do outro.

4.1 LEVANTAMENTO DE REQUISITOS

Durante a fase de levantamento de requisitos foram estimadas as características principais que o sistema de modo a subsidiar o restante do processo de desenvolvimento. Para isso, foram feitas reuniões entre os membros da equipe, *brainstorming*, entrevistas com colegas atuantes no mercado e coleta de opiniões de especialistas. O resultado de toda essa discussão pode ser visto na Tabela 4.

Tabela 4 – Levantamento de requisitos do sistema.

Requisitos	***Necessários***	***Desejáveis***
Objetivo do projeto	Elaborar um sistema embarcado reconfigurável operacional	Formalização de uma metodologia de elaboração de sistema embarcado
Aplicação	Sistemas domóticos	Sistemas Embarcados (Geral)
Utilizador	Técnicos e tecnólogos	-
Público Alvo	Atividades pedagógicas	Mercado
Utilização de Núcleo GPP	Sim	-
Restrições quanto ao fabricante do GPP	Não	-
Conversores AD/DA	Sim	10-bits
PWM	Sim	2
Real Time Clock	Sim	1
Modos de economia de energia	Sim	-
Utilização de núcleo reconfigurável	Sim	-
Restrições quanto ao fabricante do FPGA	Não	-
Quantidade mínima de *Gates*	100K	500K
Utilização de MicroBlaze	Não	Projetos futuros
Fonte de alimentação	Uma para toda a plataforma	Uma para cada núcleo, independentes
Memória Externa	Uma EEPROM para cada núcleo	Uma DDR para cada núcleo
Interface com o usuário	Botões e *LEDs* em cada núcleo	Potenciômetro
Comunicação externa	Centralizada no ARM: RS232, JTAG, SPI, I^2C, ISP, USB e Ethernet	Duas USB 2.0 e Ethernet Wireless
Restrições no fornecimento de energia	Alimentação por fonte alternada ou contínua entre 9 e 12V.	
Restrições técnicas	Funcionamento residencial ou predial	Outras aplicações de controle.
Dimensões e aparência	Não há.	Não há.
Portabilidade	Sim	-

Qualidade	Componentes e placas	-
Sustentabilidade	-	Componentes com certificação RoHS
Restrições orçamentárias	-	Menor custo, dentro dos requisitos estabelecidos sem comprometer qualidade e segurança.
Critérios de aceitação	Os dois núcleos funcionando independentemente	Os dois núcleos funcionando independentemente e integrados

Diversos outros requisitos poderiam ser levantados entretanto, considerando os objetivos deste trabalho, os requisitos ora apresentados mostraram-se suficientes para a elaboração da plataforma. A fase de levantamento de requisitos deve ser feita a extrair o máximo de informações do se idealiza que o produto final deve ou não conter. Essas informações serão guias durante todo o desenvolvimento, portanto devem ser feitas de maneira mais detalhada possível e, uma vez definidas, quanto menos susceptíveis a mudanças, melhor (Vargas, 2009).

4.2 PROJETO CONCEITUAL DO SISTEMA

Após algumas reuniões com a equipe do projeto, chegou-se a uma plataforma constituída por dois núcleos de processamento, sendo que um utilizando um microcontrolador e outro um dispositivo reconfigurável do tipo *FPGA*. Para tanto, foi definido que ambos os dispositivos deveriam funcionar independentemente um do outro ou em conjunto, conforme a necessidade do projetista. A partir disso, partiu-se para a agregação de componentes essenciais a cada um dos núcleos. O conceito elaborado é mostrado na Figura 1.

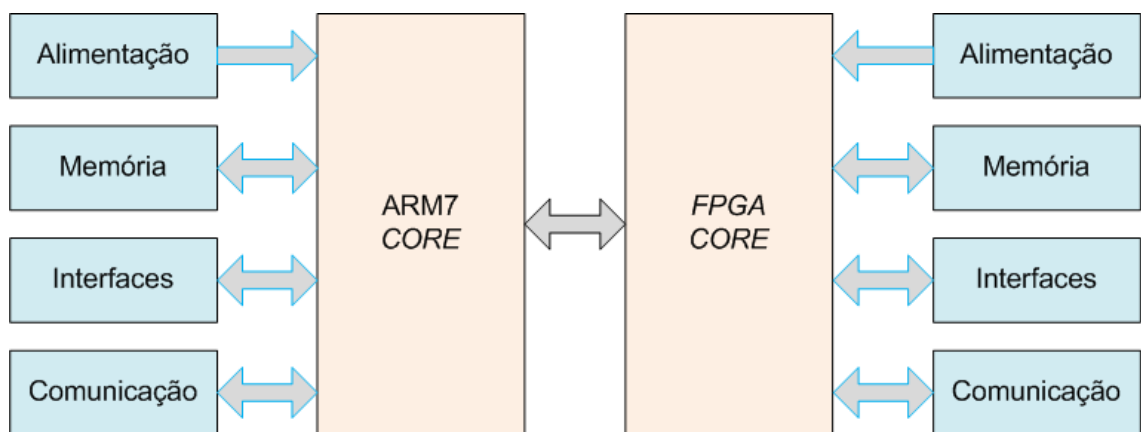



Figura 1 – Esquema conceitual da plataforma reconfigurável de controle.

4.3 ELABORAÇÃO DOS CIRCUITOS ELETRÔNICOS

Para a desenvolvimento do projeto, foi utilizado o *software* CAD/EDA Altium Designer versão 6.9. Foram utilizadas as ferramentas de elaboração de esquemático e projeto de PCI.

Durante a evolução do projeto da plataforma reconfigurável, em alguns momentos foi necessário criar componentes utilizando tanto a sua representação eletrônica como seu formato físico, que mais tarde seriam utilizados na fabricação do projeto final da PCI.

Todos os circuitos elaborados foram disponibilizados no APÊNDICE A, e explicados em mais detalhes ao longo do CAPÍTULO 4.

4.3.1 PROJETO DO MODULO DE ALIMENTAÇÃO DO ARM (ARM-ALIMENTAÇÃO)

O subsistema *ARM-Alimentação* compreende toda a parte de alimentação do microcontrolador. Este módulo constitui-se de três circuitos: (a) regulador de tensão, (b) alimentação por bateria e (c) circuito gerenciador de reinicialização do microcontrolador.

Uma das vantagens em se fazer uso de *kits* de auxilio ao desenvolvimento é utilizar o material de apoio fornecido pelo fabricante para implementar o próprio circuito. A placa fornecida em perfeito funcionamento é prova que o circuito que foi utilizado atende bem às demandas do microcontrolador. Ademais, os circuitos normalmente utilizados por desenvolvedores de *kits* de auxilio a projetos de engenharia são baseados nas recomendações dos fabricantes. Não obstante tudo isso, antes de se utilizar qualquer circuito fornecido na documentação dos *kits* de desenvolvimento referenciados, foi verificado junto à documentação do fabricante dos componentes a compatibilidade entre eles.

Todos os circuitos que se referem ao subsistema ARM, foram baseados (reproduzidos algumas vezes com, algumas sem modificação) na documentação fornecida no *kit* de desenvolvimento *PLACA McBoard ARM7.1* (LPC213X) do fabricante MOSAICO[12].

---

[12] Disponível em: http://www.mosaico.com.br/?canal=5&pg=showProduto&path=produtos&id=44. Acesso em: 09/07/2012.

4.3.1.1 PROJETO DO SISTEMA DE ALIMENTAÇÃO PARA O ARM (ARM - 9V, 5V E 3V3A)

Este circuito é o regulador de tensões que alimenta principalmente o *ARM7*. A Figura 2 mostra o esquemático do circuito regulador de tensões de 9V, 5V e 3V3 a partir de uma alimentação de entrada entre 9 e 12Volts contínua ou alternada.

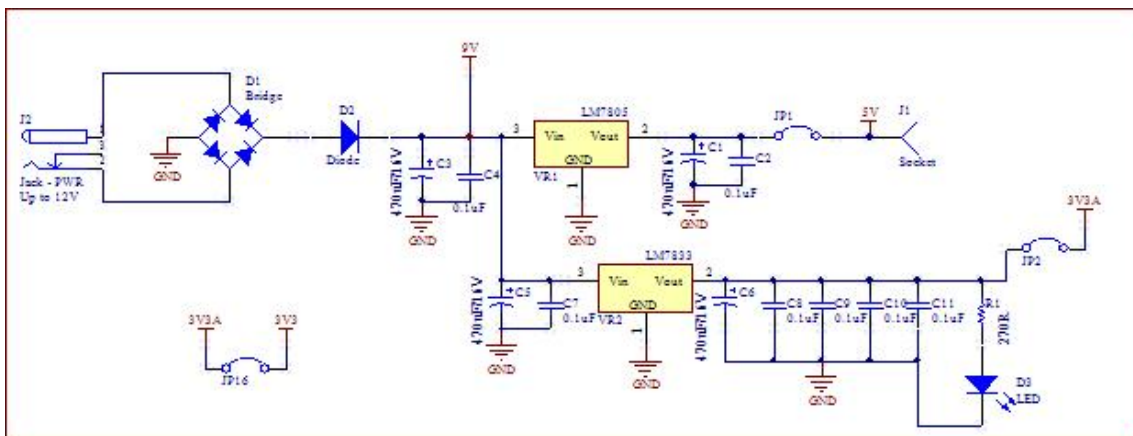



Figura 2 – Circuito regulador de tensão – 9V, 5V e 3V3 - *ARM*.

O circuito regulador de tensão é constituído, inicialmente, por um *circuito retificador de onda completa*, implementado por meio de uma ponte de diodos. Este é um circuito muito utilizado em projetos de microeletrônica, pois permite obter uma tensão contínua caso a alimentação de entrada seja alternada. Como resultado, a alimentação de entrada na placa pode ser tanto com tensão contínua como alternada. Os reguladores de tensão suportam tensão de entrada de até 25 Volts rms, entretanto para que eles trabalhem adequadamente nessa voltagem seriam necessários dissipadores de calor em cada um deles, pois seriam submetidos a temperaturas muito elevadas; podendo assim influir no funcionamento de outros componentes. Dessa forma, optou-se por limitar a tensão de entrada em 12V.

Uma vez garantida a alimentação de entrada conforme especificado, bastou inserir um regulador de tensão, LM7833, para obter a alimentação adequada ao *ARM*, 3,3 Volts, indicada no circuito como *3V3A*. Esta nomenclatura foi adotada para diferenciar da alimentação do *FPGA*, 3V3, que será descrita adiante. A representação *3V3A* remete à 3V3+*ARM*. Apesar de serem de mesmo valor, optou-se por trata-las de forma independente garantindo assim o total desacoplamento dos sistemas.

Por outro lado, também foi inserido o regulador, LM7805, para fornecer uma tensão de 5 Volts a fim de suprir eventual necessidade do usuário de uma tensão deste valor durante a utilização da placa. Além disso, essa tensão ainda auxilia o circuito de alimentação por bateria, descrito na sessão 4.3.1.2, a seguir.

Para efeitos de filtragem, antes e depois de cada regulador de tensão foram inseridos capacitores que têm como função principal filtrar e estabilizar os sinais, respectivamente de entrada e de saída de cada regulador. O dimensionamento deles seguiu a recomendação dos fabricantes disponível na documentação (*datasheet*) de cada regulador de tensão utilizado. Seguindo recomendações feitas por colegas atuantes no desenvolvimento de *PCBs* no mercado foram inseridos 4 capacitores de 0.1uF e um de 470uF, em paralelo com a saída da fonte de 3V3, de modo a reduzir a possibilidade de oscilação desta fonte uma vez que ela será utilizada para alimentar o *ARM*.

Na saída de cada regulador de tensão foi facultado ao usuário desacoplar a respectiva alimentação do restante do circuito por meio de *jumpers*. O JP1 desacopla 5V e JP2 desacopla 3V3A (3,3 direcionada ao *ARM*). Esta função é muito útil tanto no momento da montagem da placa quanto durante a utilização. Como dito anteriormente, a modularidade ou o desacoplamento dos circuitos favorece a fase de teste, pois possibilita que sejam verificados os sinais na entrada e na saída de cada subsistema, independentemente, antes que sejam aplicados ao restante da plataforma. Assim, caso a placa venha a ser danificada por qualquer razão, é possível iniciar a fase de testes a partir da alimentação, normalmente muito sensível, sem que tenha interferência do restante dos componentes (vide metodologia exposta no CAPÍTULO 3).

Na saída do regulador LM7833 foi colocado um LED (D3) indicativo para sinalizar o funcionamento da fonte de alimentação até aquele ponto. Ou seja, estando ele aceso, sabe-se, visualmente, que a placa está sendo alimentada e apta para fornecer 3,3 Volts ao restante do circuito.

4.3.1.2 PROJETO DE CIRCUITO DEDICADO À FUNÇÃO RTC (ARM – BATERIA/RTC)

O *ARM7* utilizado dá ao projetista a opção de alimentá-lo por meio de uma bateria em caso de utilização da função *RTC (Real Time Clock)* pois esta função exige uma fonte própria e ininterrupta de alimentação. Optou-se por implantar o circuito e facultar ao usuário a sua utilização. O circuito elaborado pode ser visto na Figura 3.

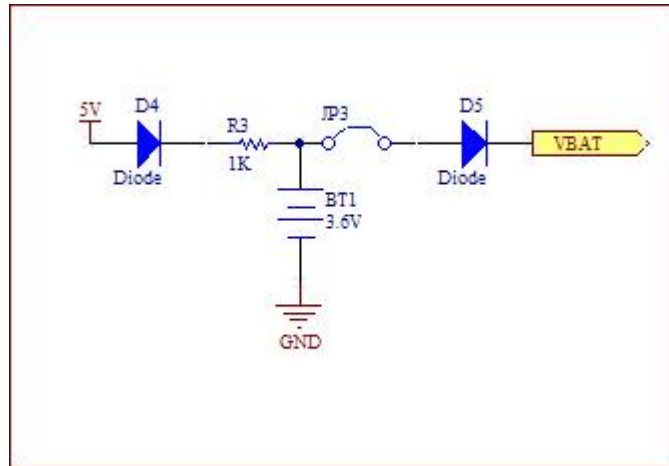



Figura 3 – Circuito de alimentação RTC utilizando bateria (*ARM-Bateria/RTC)*. Errata: a tensão deve ser de 3V3 e não 3V6 indicado no circuito.

O circuito utilizado indica uma tensão de 3V6 mas deve ser utilizada em 3V3. Optou-se por manter o circuito utilizado pois caso alguém tome os projetos utilizados para a fabricação da versão 1.1 não se perca. Destaca-se que a versão 1.2 essa anotação já foi corrigida.

4.3.1.3 SISTEMA DE RESET PARA O MÓDULO ARM (ARM-DRIVER RESET)

Por último, foi inserido, ainda no bloco *ARM - Alimentação*, um dispositivo supervisor de circuito com microcontrolador para gerenciar a reinicialização do *ARM*, em inglês, *driver reset*. Basicamente, o MCP130 é um dispositivo supervisor de voltagem que mantem o microcontrolador no estado de inicialização até que a fonte de voltagem atinja níveis estáveis de operação.

O *driver reset* também opera como um protetor de condições adversas quando a fonte de alimentação oscila abaixo dos níveis ideais de operação do microcontrolador (Microchip, [S.d.]). Além disso, ele ainda reinicia o microcontrolador em caso de perda de alimentação e também auxilia na reinicialização por opção do usuário. No circuito implementado, o simples pressionamento do botão S1 permite que o *ARM* seja reinicializado com segurança. A Figura 4 ilustra o circuito com a função de *driver reset*.

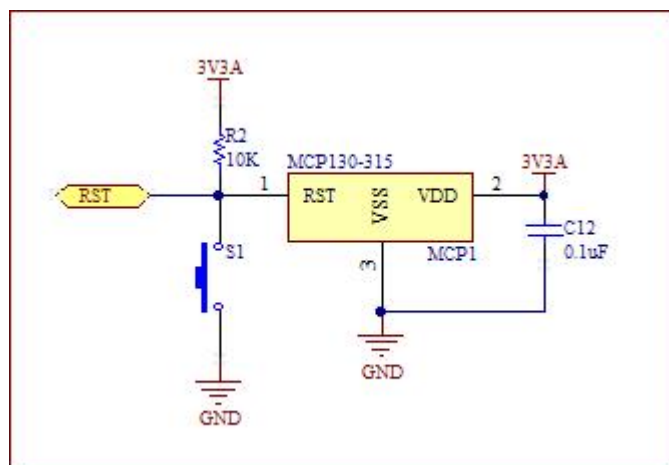



Figura 4 – Circuito gerenciador de reinicialização do *ARM* (*ARM-Driver Reset)*.

4.3.2 MÓDULO DE MEMÓRIA DO ARM (ARM - MEMÓRIA)

O próprio microcontrolador já traz em si 40k *Bytes* de memória estática de acesso aleatório – *SRAM*[13], para operações que exijam acesso aos dados ali armazenados. Os dados são disponíveis enquanto mantida sua alimentação, não precisando que as células que armazenam os bits sejam atualizadas. A *SRAM* pode ser usada tanto para armazenar código como dados. Além da *SRAM*, o microprocessador LPC2146 ainda disponibiliza 256K *Bytes* de memória *EEPROM*[14] flash[15] que, da mesma forma que a *SRAM*, pode ser usada para armazenar código e dados, mas com a característica de não volatilidade.

De modo a viabilizar projetos mais robustos, especialmente voltados à aquisição, análise, monitoramento e controle de dados, optou-se por inserir uma memória externa ao microcontrolador. Algumas alternativas foram consideradas como, por exemplo, SDRAM, DDR e DDR2 entretanto não foram implementadas. Normalmente essas memórias são feitas para trabalharem em velocidades acima de 133MHz o que fica muito acima dos 60MHz que seria o limite do *ARM7* utilizado. Além disso, fazendo uma pesquisa de mercado pelos

---

[13] Em inglês, *Static Random Access Memory*.

[14]Em ingês, *Electrically Erasable Programmable Read-Only Memory,* ou simplesmente, *EEPROM*. É uma memória que tem a característica de ser não volátil, preserva seu conteúdo mesmo na ausência de alimentação, com suporte a múltiplas escritas e leituras em partes ou no todo.

[15] Memória *flash* é um tipo específico de memória *EEPROM* que usa mecanismos para gravar e apagar em blocos, geralmente de 512 bytes, por meio da aplicação de um campo elétrico. Esse acesso aos dados em blocos faz com que ela realize operações mais rapidamente que as tradicionais *EEPROMs*, que normalmente acessam os dados *byte* a *byte*.

diversos Kits baseados nesse microprocessador, percebeu-se que não é usual o uso de memórias desse tipo para plataformas desse porte.

Assim, foi escolhida uma *EEPROM* com capacidade de armazenamento de 1 *Mega bytes*. Para tanto, foi selecionado o *CI* 24LC1025 da *Microchip Technology* e sua ligação foi feita conforme mostrado na Figura 5.

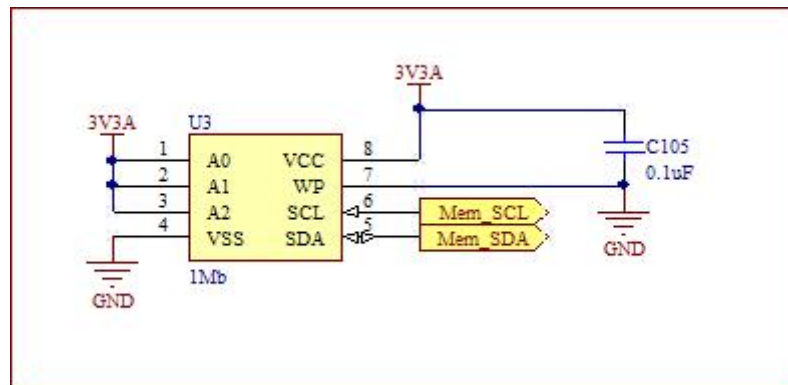



Figura 5 – Circuito para memória externa do *ARM (ARM-Memória)*.

4.3.3 PROJETO DE INTERFACES PARA O ARM (ARM-INTERFACES)

Outro módulo que foi criado foi o de interfaces com o usuário. Este módulo congrega circuitos dedicados à interação com o usuário ou desenvolvedor. Durante a fase de elaboração de novos projetos, por vezes são necessários testes de sinais, entrada e saída de dados que de maneira simples podem ser implementados na placa, auxiliando assim o desenvolvedor. Há ainda projetos que dependem de entrada e retorno ao próprio usuário, e quando não estão implementadas no próprio circuito, exigirá um trabalho extra do desenvolvedor para montar e compatibilizar componentes extras, visando suprir as necessidades do projeto.

4.3.3.1 INTERFACE DE ENTRADAS DIGITAIS (ARM-PUSH BUTTON)

Foram elaborados dois modos para entrada de dados: (a) botões (caracterizando entradas digitais) e (b) potenciômetro (caracterizando uma entrada analógica). Para o caso do primeiro tipo de entrada, optou-se por utilizar cinco botões para interface de entradas digitais. Esses botões podem ser utilizados independentemente ou em conjunto conforme a necessidade de entrada de parâmetros. A Figura 6 mostra o circuito utilizado.

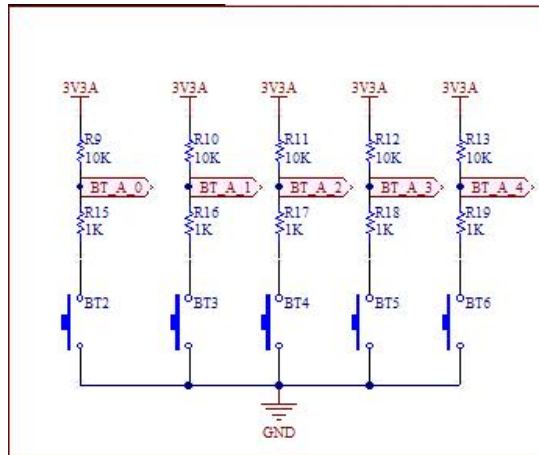



Figura 6 – Circuito de interface de entrada com botões (*ARM-Push Buttons)*.

A nomenclatura utilizada para cada botão, *BT_A_X*, corresponde a botões ligados ao *ARM* seguido de seu respectivo número sequencial, sendo que cada botão está ligado a um pino correspondente no *ARM*, conforme a Tabela 5.

Tabela 5 – Correspondência entre botões de interface e pinos no *ARM*.

Botão	***Pino no ARM***	***PORT***
BT2	15	P0.30
BT3	44	P1.21
BT4	40	P1.22
BT5	36	P1.23
BT6	28	P1.25

4.3.3.2 INTERFACE DE ENTRADA ANALÓGICA (ARM-POTENCIÔMETRO)

Para a entrada analógica foi inserido um potenciômetro de 10KΩ de modo a dar mais versatilidade à placa com uma entrada variável, multivalorada. A Figura 7 mostra o circuito utilizado para agregar o potenciômetro à plataforma. A nomenclatura *POT_A* corresponde a potenciômetro ligado ao *ARM*.

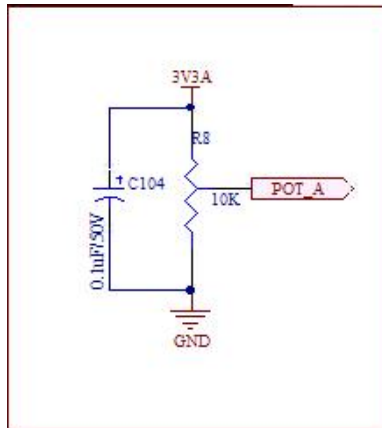



Figura 7 – Circuito do potenciômetro ligado ao *ARM (ARM-Potenciômetro)*.

4.3.3.3 INTERFACE DE SAÍDAS DIGITAIS DO ARM (ARM-LEDs)

Além das interfaces de entrada, foi implementada ainda uma interface de saída composta por cinco *LEDs*. Interfaces de saída são importantes especialmente durante a fase de desenvolvimento de novos projetos (*debugging*), pois servem como retorno visual em testes de funcionamento de partes de algoritmos. Além disso, pode servir também para notificação, alerta ou sinalização ao usuário. A Figura 8 mostra o circuito utilizado para a interface com *LEDs*.

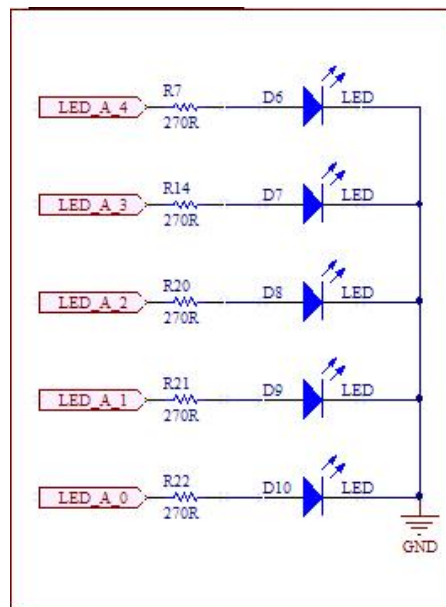



Figura 8 – Circuito de interface de saída com *LEDs* (*ARM-LEDs)*.

A nomenclatura utilizada para cada *LED*, *LED_A_X*, corresponde a *LED* ligado ao *ARM* seguido de seu respectivo número sequencial. Cada *LED* está ligado a um pino correspondente no *ARM*, conforme a Tabela 6.

Tabela 6 – Correspondência entre *LEDs* de interface e pinos no *ARM*.

LED	*Pino no ARM*	*PORT*
D6	48	P1.20
D7	4	P1.19
D8	8	P1.18
D9	12	P1.17
D10	16	P1.16

4.3.4 PROJETO DOS SISTEMAS DE COMUNICAÇÃO LIGADOS AO ARM (ARM-COMUNICAÇÃO)

Por último, foi implementado um módulo voltado exclusivamente à comunicação externa do *ARM* utilizando todas as interfaces disponíveis na estrutura do *ARM* selecionado, são elas: (a) *SPI*, (b) *I^2C*, (c) *BSD/ISP*, (d) *JTAG*, (e) *USB* e (f) *RS232 half e full duplex*.

4.3.4.1 MÓDULO DE COMUNICAÇÃO DO ARM (ARM-SPI)

O *Serial Peripheral Interface – SPI* é um protocolo síncrono[16] de interface serial[17] que opera em modo *full duplex*[18]. Pode ser usado para manipular múltiplos dispositivos mestres e escravos estando eles conectados em um determinado barramento. Por ser uma comunicação *duplex*, apenas um único mestre (*master*) ou escravo (*slave*) pode se comunicar pela interface durante a transferência de dados. Em outras palavras, o protocolo *SPI* não permite o endereçamento, sendo que a comunicação só pode ser feita entre dois pontos, sendo um deles o *master* e outro o *slave* (Souza; Lavinia, 2002). A Figura 9 mostra o circuito utilizado para disponibilizar a comunicação *ARM – SPI*.

---

[16] Numa comunicação síncrona, cada bloco de informação é transmitido e recebido num instante de tempo bem definido e conhecido pelo transmissor e receptor, ou seja, estes têm que estar sincronizados. Para se manter esta sincronia, é transmitido periodicamente um bloco de informação que ajuda a manter o emissor e receptor sincronizados, denominado *clock*. Diferentemente da comunicação síncrona, na assíncrona cada bloco de dados inclui um bloco de informação de controle, chamado *flag*, para que se saiba exatamente onde começa e termina o bloco de dados e qual a sua posição na sequência de informação transmitida.

[17] Comunicação serial é o processo de enviar dados *bit* a *bit*, sequencialmente, num canal de comunicação ou barramento.

[18] *Duplex* é um sistema de comunicação composto por dois interlocutores que podem comunicar entre si em ambas direções. Uma comunicação é dita *full duplex*, as vezes chamada apenas de *duplex*, quando os dispositivos transmissor e receptor podem trocar dados simultaneamente em ambos os sentidos, utilizando uma transmissão bidirecional.

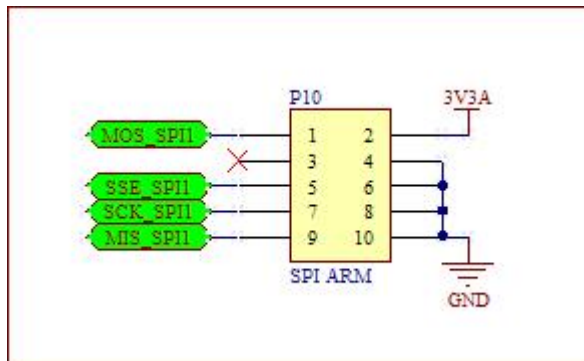



Figura 9 – Circuito de comunicação para interface *ARM-SPI***.**

Essa comunicação é feita utilizando quatro vias:

a) *Clock*: trata-se da via de *clock*, que pode ser entrada (*slave*) ou saída (*master*). No microcontrolador 24LC2146 é chamado de SCK e está localizado no pino 27 (P0.4).

b) MISO - *Master In Slave Out*: este pino servirá como entrada caso o microcontrolador esteja programado para funcionar como *master*. Mas ele também funciona como saída de dados caso o microcontrolador esteja como funcionando como *slave*. Esta função corresponde ao pino 29 (P0.5) no microcontrolador.

c) MOSI - *Master Out Slave In*: este pino servirá como saída caso o microcontrolador esteja programado para funcionar como *master*. Mas ele também funciona como entrada de dados caso o microcontrolador esteja como *slave*. Esta função corresponde ao pino 30 (P0.6) no microcontrolador.

d) SSEL - *Slave Select*: é um sinal utilizado para seleção do *slave*. Com este recurso pode-se montar uma rede de comunicação com um *master* e vários *slaves,* desde que o *master* controle individualmente os pinos SSEL de cada um dos *slaves*. Esse controle deverá ser implementado manualmente no *software* de modo a garantir que somente um *slave* esteja ativado de cada vez, evitando conflitos. Esta função corresponde ao pino 31 (P0.7) no microcontrolador.

O microcontrolador utilizado já traz implementado em si duas interfaces de comunicação: SPI0 e SPI1. Optou-se então por utilizar uma delas, SPI0, para a comunicação externa e a outra, SPI1, para uma eventual comunicação com o *FPGA*. Foi disponibilizado um conector 5X2 exclusivo para a comunicação *SPI externa*. Ele está identificado na placa como P10. A Tabela 7 disponibiliza o sinal correspondente a cada pino para a comunicação externa *ARM-SPI*.

Tabela 7 – Correspondência entre pinos e sinais da comunicação *ARM-SPI*.

Pino	Sinal	ARM
1	MOSI0	30 (P0.6)
2	3V3A	
3	Não Conectado	
4	GND	
5	SSEL0	31 (P0.7)
6	GND	
7	SCK0	27 (P0.4)
8	GND	
9	MISO0	29 (P0.5)
10	GND	

4.3.4.2 MÓDULO DE COMUNICAÇÃO DO ARM (ARM-JTAG)

Em 1985, um consórcio entre os maiores fabricantes mundiais de circuitos integrados foi formado e recebeu o nome de *The Joint Test Access Group*, ou *JTAG*. O trabalho do grupo era focado em desenvolver padrões de testes para circuitos integrados e placas de circuito impresso. O trabalho desenvolvido pelo *JTAG* foi de tamanho reconhecimento que acabou sendo aprovado pelo Instituto de Engenheiros Eletricistas e Eletrônicos – IEEE como o padrão *IEEE Std. 1149.1*: *Standard Test Access Port and Boundary-Scan Architecture,* ou simplesmente o padrão *JTAG*. A partir daí o *JTAG* se tornou uma interface largamente utilizada, especialmente em projetos envolvendo microcontroladores, para testes e depuração de sistemas embarcados (Fernandez Filho, 2009).

O padrão *JTAG* básico faz uso de cinco vias de comunicação, e dois outros sinais também podem ser utilizados (*RTCK e JTRST*):

a) *TDI (Test Data In):* entrada de dados do dispositivo*.*

b) *TDO (Test Data Out):* dados lidos do dispositivo.

c) *TMS (Test Mode Select):* controla a máquina de estados *JTAG.*

d) *TCK (Test Clock):* sinal de *clock.*

e) *RTCK (Returned Test Clock):* sinal opcional. Usado como retorno do *clock* para sincronização.

f) *TRST (Test Reset):* sinal opcional para *reset* assíncrono do dispositivo.

g) *JTRST (Test Reset): sinal opcional para reset assíncrono do dispositivo.*

Este protocolo utiliza comunicação serial, vez que faz uso de apenas uma via para cada sentido de transmissão, *TDI* e *TDO*. Os outros sinais – *TMS, nTCK* e opcionalmente *nTRST* – formam o chamado controlador *TAP* ou *Test Access Port controller*. A Figura 10 ilustra uma cadeia de dispositivos utilizando *JTAG*.

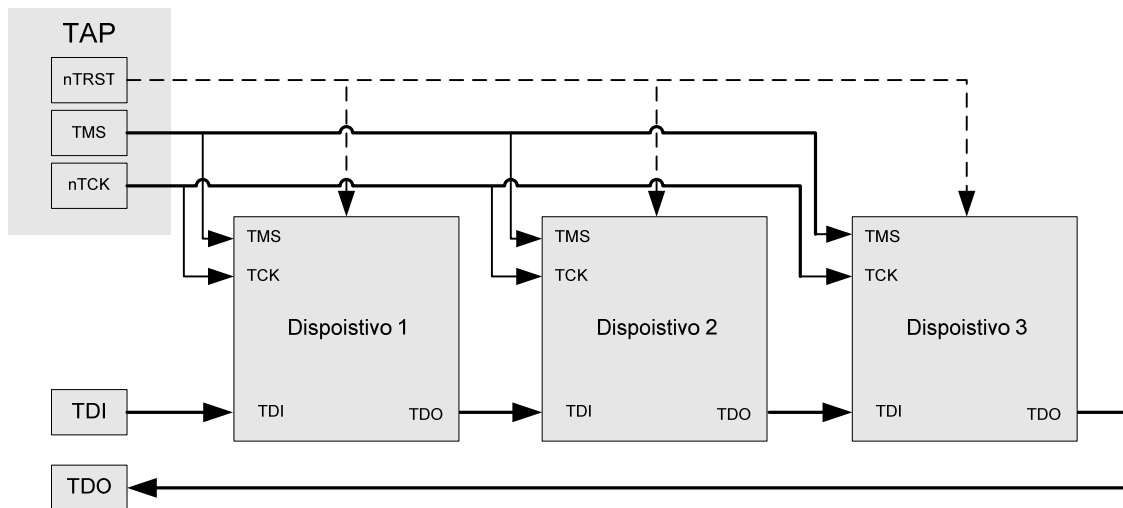



Figura 10 – Gráfico representativo de interconexão de dispositivos utilizando *JTAG*.

O controlador *TAP* é responsável pela seleção interna de registradores e por toda a máquina de estados que controla o fluxo de dados na placa, quando em modo *JTAG*. Nesse contexto, o sinal *TCK* é responsável pela sincronia interna da máquina de estados, *TMS* é o sinal que determina o próximo estado e *nTRST*, quando disponível, é o sinal que pode reiniciar a máquina de estados *JTAG*. O estado inicial de operação é o estado *test logic reset*, ou TLR, que é atingido após 5 ciclos de *TCK* com *TMS* em nível lógico alto. Enquanto nesse estado, se *TMS* estiver em nível lógico baixo em um dado pulso de *TCK*, a máquina de estados entrará em *Run-Test/Idle*. Os próximos estados serão, então, um conjunto formado por (a) seleção, (b) captura, (c) deslocamento, (d) pausa, (e) fim e (f) atualização de registradores de dados ou de registradores de instrução.

Utilizando o material de referência do *kit* utilizado em laboratório, bem como recomendações do fabricante do microcontrolador, foi montado um circuito para disponibilizar a interface *JTAG*, como pode ser visto na Figura 11.

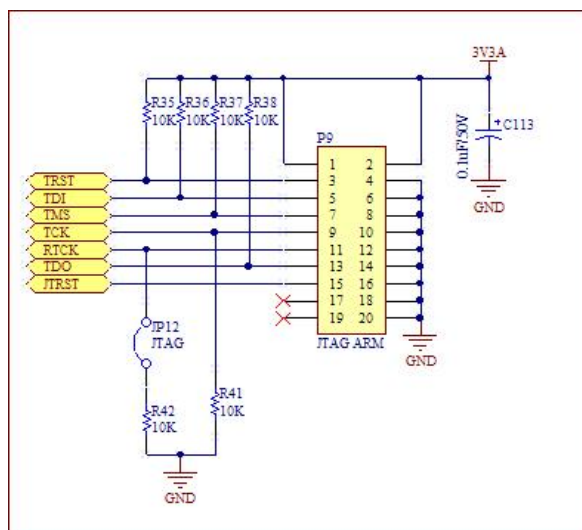



Figura 11 – Circuito de comunicação, interface *JTAG* (*ARM-JTAG)*.

Foi disponibilizado um conector 10X2 exclusivo para a comunicação *JTAG*. Ele está identificado na placa como P9. A Tabela 8 disponibiliza o sinal correspondente a cada pino para a comunicação externa *ARM-JTAG*.

Tabela 8 – Correspondência entre pinos e sinais no conector *ARM-JTAG*.

Pino	*Sinal*	*ARM*
1	3V3A	
2	3V3A	
3	TRST	20 (P1.31)
4	GND	
5	TDI	60(P1.28)
6	GND	
7	TMS	52 (P1.30)
8	GND	
9	TCK	56 (P1.29)
10	GND	
11	RTCK	24 (P1.26)
12	GND	
13	TDO	64 (P1.27)
14	GND	
15	JTRST	32 (P1.24)
16	GND	
17	NC	
18	GND	
19	NC	
20	GND	

4.3.4.3 MÓDULO DE COMUNICAÇÃO DO ARM (ARM-ISP/IAP)

Basicamente, existem dois métodos tradicionais para se programar um *ARM* da família *LPC*: (a) utilizar a interface *JTAG* para descarregar o *software* já

compilado ou então (b) utilizar o a interface *ISP* – in system programming – que utiliza apenas 3 vias, *MISO, MOSI* e *SCK*.

O ISP é uma interface de programação ou reprogramação de todas as memórias não voláteis do processador em operação, utilizando um *software* de controle durante a inicialização (*boot loader*). Isso pode ser feito mesmo quando o dispositivo já se encontra instalado no produto do consumidor final, ou seja, é uma forma de viabilizar a atualização de *firmware* mesmo na fase pós-venda. O *boot loader* também habilita o modo *IAP - in application programming,* que, de forma muito parecida com o ISP, permite apagar e escrever na memória *flash* do *ARM* mas, nesse caso, utilizando-se o código do aplicativo do usuário final.

Para habilitar o modo *ISP/IAP* no microcontrolador, caso ele dê suporte a esse modo, basta forçar o controlador do *ISP/IAP*, no caso chamado de *boot loader select* – BSL, para o nível baixo, no caso 0 Volts, enquanto o dispositivo é inicializado. Ao iniciar em modo *ISP/IAP* o programador poderá atualizar o dispositivo transferindo assim o código para a memória *flash* interna, por exemplo. A partir daí, ao reinicializar o dispositivo com o *BSL* em nível alto, ele já estará funcionando com o *firmware* atualizado.

O circuito utilizado para habilitar o modo *ISP/IAP* está descrito na Figura 12. Para utilizar o modo *ISP/IAP* basta inserir um *jumper* no terminal P12 da placa que isso forçará o *BSL*, que no *ARM* corresponde ao pino 41 – P0.14, assuma o controle da inicialização do sistema. Para executar um programa basta retirar o *jumper* e reinicializar o sistema.

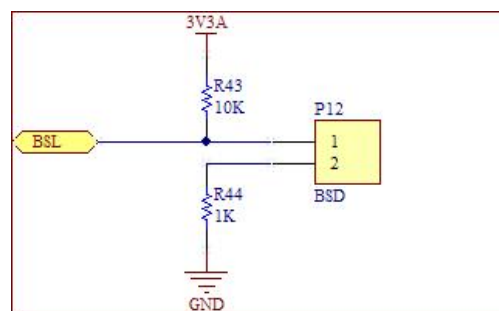



Figura 12 - Circuito de comunicação para a interface *ARM-ISP/IAP*.

A Tabela 9 apresenta o resumo da função do *jumper* em P12 na placa para a programação ou execução de código.

Tabela 9 – Função do jumper em P12, *ARM-ISP/IAP*.

Jumper	***Fechado***	***Aberto***
P12	Modo ISP/IAP: programação	Execução de programa

4.3.4.4 MÓDULO DE COMUNICAÇÃO DO ARM (ARM-I2C)

Em inglês, esse protocolo é denominado *Inter-Integrated Circuit – I^2C*, em tradução literal, seria *entre circuitos integrados*. Conforme Souza e Lavinia (2002), o padrão de comunicação *I^2C* possui características bem diferentes do *SPI*. Este protocolo continua com o conceito de *master/slave*; entretanto, ele permite o endereçamento de diversos pontos da rede por meio das próprias vias de comunicação. Com isso é possível ter um *master* com diversos *slaves* sem a necessidade de pinos de controle adicionais. De fato, é possível também a estruturação de uma rede com diversos mestres.

A comunicação é feita somente em duas vias: *clock* e *data*. Como também se trata de uma comunicação síncrona, a via de *clock* continua sendo indispensável. Quanto aos dados, transitam em uma única via, tanto para a transmissão quanto para a recepção.

Por outro lado, tanto o *master* quanto o s*lave* podem transmitir e receber dados, mas o controle é sempre do *master*. Para evitar conflitos na via de dados (*serial data input/output - SDA*), os pinos são chaveados como entrada ou saída (impõe somente o nível baixo, forçando o GND), conforme a necessidade imposta pelo protocolo. Por isso, foram utilizados resistores *pull-up*[19] de 2kΩ. Valores mais baixos (até 1kΩ) podem ser utilizados para velocidades de transmissão mais elevadas ou então no caso de uma rede com muitos periféricos.

O *clock* (*Serial Clock - SCL*) deve ser controlado exclusivamente pelo *master*. A questão é que o I^2C compartilha uma única via de dados e pela possibilidade de existir mais de um *master*, o *clock* necessário para a resposta do *slave* deve ser emitido no momento correto, evitando assim conflitos no processamento da informação. Para que isso seja possível, a via de *clock* também é alterada como entrada e saída durante a transmissão, permanecendo, em determinados momentos, em um estado flutuante. Para que isso não ocorra, também foi

[19] Resistores *pull-up* são usados para garantir que entradas para sistemas lógicos se ajustem em níveis lógicos esperados se os dispositivos externos são desconectados. A ideia de um resistor *pull-up* é que ele “puxe” (pull) fracamente a tensão do condutor a que ele está conectado para o nível lógico alto, no caso, 3,3V. Contudo, o resistor é intencionalmente fraco, com alta-resistência, de modo que caso qualquer outro mecanismo que puxe a tensão do condutor fortemente para 0V, a tensão irá para 0V, ou o nível lógico baixo que se esteja trabalhando.

instalado um resistor de *pull-up* para evitar esse estado flutuante, e não gerar assim problemas no protocolo. Normalmente, utiliza-se resistores entre 1kΩ e 10kΩ tanto para o pino de dados quanto para o de *clock*. Optou-se, no presente caso, por utilizar resistores de 2kΩ em ambos os pinos para garantir sincronia e possibilidade de altas velocidades de transmissão de dados (em torno de 400kbit/s).

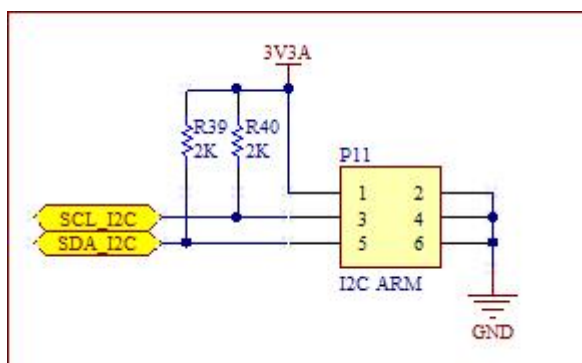



Figura 13 – Circuito de comunicação para a interface *ARM-I^2C*.

O microcontrolador utilizado já traz implementado em si duas interfaces de comunicação *I^2C*. Optou-se por utilizar uma delas, *I^2C0* para a comunicação externa e a outra, *I^2C1*, para uma eventual comunicação com o *FPGA*. Foi disponibilizado um conector 3X2 exclusivo para a comunicação *I^2C* externa. Ele está identificado na placa como *P11*. A Tabela 10 correlaciona o sinal correspondente a cada pino para a comunicação externa *ARM-I^2C*.

Tabela 10 – Correspondência entre pinos e sinais da comunicação externa *ARM-I^2C*.

Pino	***Sinal***	***ARM***
1	3V3A	
2	GND	
3	SCL0	22 (P0.2)
4	GND	
5	SDA0	26 (P0.3)
6	GND	

4.3.4.5 MÓDULO DE COMUNICAÇÃO DO ARM (ARM-RS232)

Segundo Canzian ([S.d.]), RS é uma abreviação de *Recommended Standard,* ou padrão recomendado. Ele relata uma padronização de uma interface comum para comunicação de dados entre equipamentos, criada no início dos anos 60, por um comitê conhecido atualmente como *Electronic Industries Association* - EIA. Naquele tempo, a comunicação de dados compreendia a

troca de dados digitais entre um computador central (*mainframe*) e terminais de computador remotos, ou entre dois terminais sem o envolvimento do computador. Estes dispositivos poderiam ser conectados através de linha telefônica, e consequentemente necessitavam de um *modem* em cada lado para fazer a decodificação dos sinais. Dessas descobertas nasceu o padrão RS232. Ele especifica (a) as tensões, (b) temporizações e funções dos sinais, (c) um protocolo para troca de informações e (d) as conexões mecânicas.

Conforme Machado Junior (1999), a interface entre um computador ou um terminal e um modem é um exemplo de um protocolo da camada física, pois ele deve especificar em detalhe a interface mecânica, elétrica, funcional e procedimental. Estes sinais são especificados por um conector de 25 pinos tipo-D. A conexão mecânica é padronizada pela *International Standart Organization - ISO*, por meio de um conector padrão denominado DB-25P.

Como forma de se otimizar o espaço na placa e maximizar as suas funcionalidades, optou-se por implementar a interface RS232 utilizando apenas as vias essenciais para a transmissão de dados. Esta é uma forma bastante difundida entre desenvolvedores de sistemas embarcados. Duas são as vias essenciais de utilização do RS232, uma para transmissão e outra para recepção de sinais. Observa-se assim, que a RS232 é um protocolo assíncrono.

A Figura 14 ilustra o circuito que foi utilizado para disponibilizar duas interfaces de comunicação RS232 *Half Duplex* (DB1 e DB2) ou uma interface *Full Duplex, no DB2 ao fechar os jumpers JP10 e JP11*.

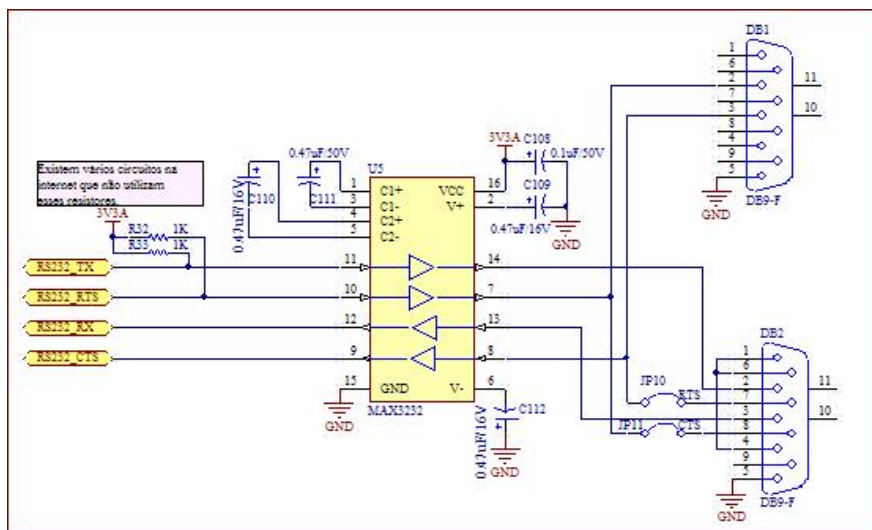



Figura 14 – Circuito de comunicação para a interface RS232 – *ARM*.

Cada UART[20] disponibiliza um canal para transmissão e um para recepção de dados. Utilizando esse par de canais é possível usar a *RS232* na configuração *half duplex*, ou seja, os dispositivos transmissor e receptor só podem trocar dados de maneira intercalada, utilizando uma transmissão unidirecional. A Tabela 11 mostra os sinais utilizado na comunicação RS232, *half duplex*.

Tabela 11 – Correspondência entre pinos e sinais da comunicação RS232 (*half duplex*) – *ARM*.

Sinal	***Nome***	***Função***	***ARM (PORT)***	***DB9-F***
TxD0	*Transmitter output Data*	Transmissão de dados	19 (P0.0)	3 (DB2)
RxD0	*Receiver Input Data*	Recepção de dados	21 (P0.1)	2 (DB2)
TxD1	*Transmitter output Data*	Transmissão de dados	33 (P0.8)	3 (DB1)
RxD1	*Receiver Input Data*	Recepção de dados	34 (P0.9)	2 (DB1)

Para cada par de sinais, *TxD* e *RxD,* foi utilizado um circuito para montar uma saída *RS232 Half Duplex*. Cada saída dessas corresponde a um conector DB09, identificados na placa como DB1 e DB2. Entretanto, foi implementada ainda uma estrutura que dá ao usuário a opção de usar uma RS232 *Full Duplex* apenas no conector DB2 em vez de duas *Half Duplex*. A conexão *Full Duplex* requer duas vias para transmissão e duas vias para recepção. Para tanto, as vias *RxT1* e *TxD1* foram compartilhadas no DB2 por meio de dois jumpers, JP10 e JP11, respectivamente. Assim a linha de dados *RxT0* passa a funcionar como *CTS* e o *TxD0* passa a funcionar como *RTS*, conforme Tabela 12.

Tabela 12 – Correspondência entre pinos e sinais da comunicação RS232 (*full duplex*) – *ARM*.

Sinal	***Nome***	***Função***	***ARM (PORT)***	***DB9-F***
TxD0	*Transmitter output Data*	Transmissão de dados	19 (P0.0)	3 (DB2)
RxD0	*Receiver Input Data*	Recepção de dados	21 (P0.1)	2 (DB2)
RTS	*Request to Send output*	2º via de transmissão	33 (P0.8)	7(DB2)
CTS	*Clear to Send*	2ª via de recepção	34 (P0.9)	8(DB2)

A Tabela 13 mostra, resumidamente, a função dos jumpers JP10 e JP11 para o circuito RS232.

Tabela 13 – Correspondência entre pinos e sinais da comunicação RS232 (*full duplex*) – *ARM*.

Jumper	***Fechado***	***Aberto***
JP10	*RxD0 se torna RTS no DB2 – RS232 Full Duplex*	RxD0 apenas para a função de receptor DB1 - RS232 *Half Duplex*
JP11	*TxD0 se torna CTS no DB2 – RS232 Full Duplex*	TxD0 apenas para a função de transmissor DB1 - RS232 *Half Duplex*

[20] *Universal Asynchronous Receiver/Transmitter*: parte dos circuitos integrados capaz de traduzir comunicação serial e paralela.

4.3.4.6 MÓDULO DE COMUNICAÇÃO DO ARM (ARM-USB)

Conforme Medeiros Júnior; Silva (2006) o barramento *Universal Serial Bus – USB* tem, desde seu surgimento (1995), se tornado o padrão da microinformática. O seu desenvolvimento partiu da colaboração de diversas empresas líderes de mercado nos segmentos de *software* e *hardware*. A sua popularidade pode ser atribuída, principalmente, às seguintes características:

a) *Facilidade do uso*: na maioria dos casos não há a necessidade de configurações no *hardware* para que ele funcione, basta conectá-lo a um *host* e o mesmo se encarrega de reconhecer a conexão elétrica do dispositivo.

b) *Velocidade*: o barramento não se torna o gargalo da comunicação entre o dispositivo e um microcomputador para o caso em questão, por exemplo.

c) *Confiabilidade*: os erros são raros de ocorrer e quando ocorrem são automaticamente corrigidos pelos algoritmos implantados com o padrão.

d) *Flexibilidade*: uma grande variedade de periféricos e dispositivos pode ser utilizada utilizando para comunicação o padrão *USB*.

e) *Preço*: os custos baixos dos componentes não são empecilhos para os fabricantes construírem a interface com seus produtos e criam uma vantagem para os consumidores finais.

f) *Praticidade de uso*: os desenvolvedores, na maioria das vezes, não necessitam de escrever drivers de baixo nível para os periféricos que utilizam o barramento.

Essa única interface permite a utilização de até 127 dispositivos diferentes através da ligação de hubs. O *USB* suporta barramentos com três velocidades de acordo com a versão da especificação. A versão 1.0 (*low-speed*) atinge a taxa de 1,5Mbps. A versão 1.1 (*full-speed*) atinge a taxa de 12Mbps. Por fim, a versão 2.0 (*high-speed*) atinge a taxa de 480Mbps. No caso, o LPC2146 já disponibiliza a versão 2.0.

A configuração utilizada no circuito foi extraída dos manuais de utilização do LPC2146 para utilização *USB*. A Figura 15 mostra o esquema do circuito para utilização da Comunicação *USB*.

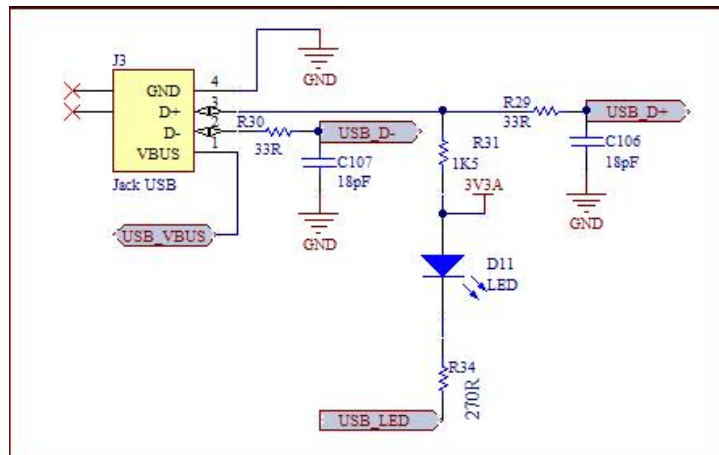



Figura 15 – Circuito de comunicação utilizando interface *ARM-USB*.

4.3.5 MÓDULO PRINCIPAL DO ARM (ARM-CORE)

O módulo *ARM-CORE* agrega apenas os componentes puramente vinculados ao *ARM*, essenciais para seu pleno funcionamento. A Figura 16 mostra o esquemático completo dos elementos que compõem este módulo.

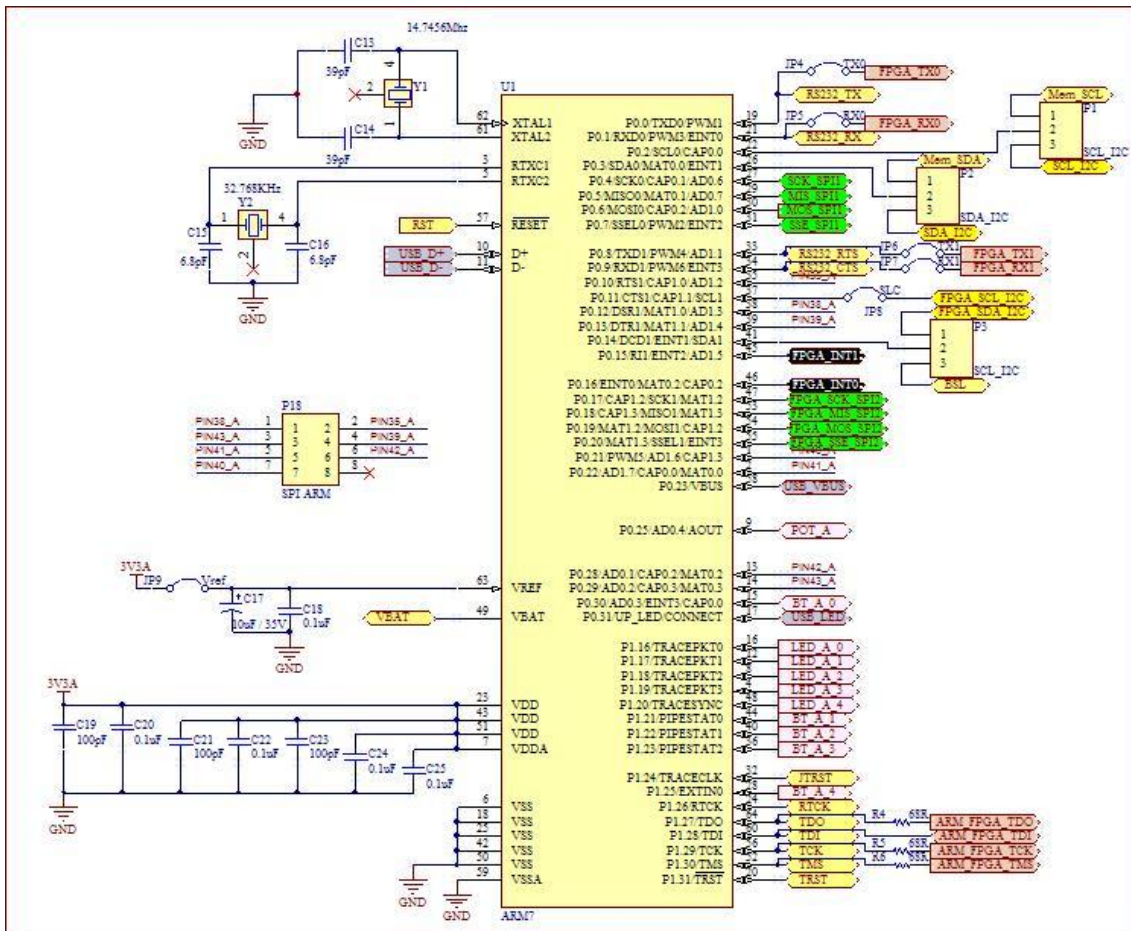



Figura 16 – Esquemático completo do módulo base *ARM7*.

Basicamente, foram inseridos dois osciladores, um para o *clock* interno e o outro para o circuito *RTC*. Além deles, alguns capacitores foram utilizados próximos aos pinos de alimentação para servirem de filtro contra ruídos.

Fazendo uma descrição seguindo o sentido anti-horário, tem-se em a sequência dos pinos e forma de utilização conforme Tabela 14.

Tabela 14 – Descrição dos pinos, função do *ARM7* e respectiva utilização na placa.

PINO	***NOME***	***TIPO***	***DESCRIÇÃO***
62	*XTAL1*	I	Entrada para oscilador e circuitos de geradores de *clock* interno. Utilizou-se um oscilador de 14.7456Mhz conforme documentação e circuitos de referência.
61	*XTAL2*	O	Saída do amplificador de oscilação. Utilizou-se um oscilador de 14.7456Mhz conforme documentação e circuitos de referência.
3	*RTCX1*	I	Entrada para o circuito oscilador *RTC* (*real time clock*). Utilizou-se um oscilador de 32.768KHz conforme documentação e circuitos de referência.
5	*RTCX2*	O	Saída para o circuito oscilador RTC (*real time clock*). Utilizou-se um oscilador de 32.768KHz conforme documentação e circuitos de referência.

57	*RESET*	I	Entrada de *reset* externa. Um sinal baixo neste pino reinicializa o dispositivo. Isso faz com que as portas de entrada e saída bem como periféricos assumam seu estado padrão e o processador é iniciado a partir de seu endereçamento 0. *TTL* com histerese, tolerância 5V. Este pino recebe o sinal vindo do bloco *ARM – Driver Reset*.
10	*D+*	I/O	Canal bidirecional D+ de dados *USB*. Este pino recebe o sinal vindo do bloco *ARM - USB*.
11	*D-*	I/O	Canal bidirecional D- de dados *USB*. Este pino recebe o sinal vindo do bloco *ARM – USB*.
63	*VREF*	I	Referência para conversores *A/D* e *DAC*. Nominalmente, deve ser o mesmo que *Vdd* mas pode ser isolado para evitar ruído e erros. Este pino deve estar semelhante ao *Vdd* caso os conversores *A/D* (*analog to digital*) e *D/C* (*digital to analog*) não estiverem sendo utilizados. Por opção de projeto, este pino foi alimentado com a tensão 3V3A (*ARM-Alimentação*) por padrão, mas foi inserido um jumper (JP9) a fim de possibilitar ao usuário utilizar outra alimentação conforme desejar. Próximo ao pino, foram inseridos capacitores com o propósito de filtro de sinais.
49	*VBAT*	I	Fonte de alimentação do RTC. Recebe a saída do – Circuito de alimentação RTC utilizando **bateria (*ARM-Bateria/RTC).*** ARM-*Bateria/RTC*.
23, 43, 51.	*Vdd*	I	Entrada de alimentação 3V3A. Recebe a saída do circuito do módulo *ARM-Alimentação*, tensão 3V3A.
7	*Vdda*	I	Entrada de alimentação analógica. Nominalmente, deve ser a mesma que *Vdd* mas pode ser isolado para evitar ruído e erros. Esta tensão serve para alimentar os *ADC* e *DAC*. Este pino deve estar amarrado ao *Vdd* caso os conversores *A/D* (*analog to digital*) e *D/C* (*digital to analog*) não estiverem sendo utilizados. Por opção de projeto, este pino foi vinculado ao *Vdd*.
6, 18, 25, 42, 50.	*Vss*	I	Terra (*GND*): referência de nível baixo ou 0V. Recebe a referência do circuito *ARM-Alimentação*.
59	*Vssa*	I	Terra (*GND*) analógico. Nominalmente, deve ser o mesmo que *Vss* mas pode ser isolado para evitar ruído e erros. Este pino deve estar amarrado ao *Vss* caso os *ADC* e *DAC* não estiverem sendo utilizados. Por opção de projeto, este pino foi vinculado à *Vss*.
20	*P1.31/TRST*	I/O	P1.31: Pino de propósito geral, entrada e saída digital.
		I	TRST: *Test Reset* para da interface *JTAG*. Este pino foi utilizado com este propósito, para *JTAG* externo.
52	*P1.30/TMS*	I/O	P1.30: Pino de propósito geral, entrada e saída digital.

		I	TMS: *Test Mode Select* para da interface *JTAG*. Este pino foi utilizado com este propósito para *JTAG* externo e também é compartilhado com o *FPGA*.
56	*P1.29/TCK*	I/O	P1.29: Pino de propósito geral, entrada e saída digital.
		I	TCK: *Test Clock* para interface *JTAG*. Este *clock* deve estar abaixo de 1/6 do *clock* da CPU (*CCLK*) para o correto funcionamento da interface *JTAG*. Este pino foi utilizado com este propósito para *JTAG* externo e também é compartilhado com o *FPGA*.
60	*P1.28/TDI*	I/O	P1.28: Pino de propósito geral, entrada e saída digital.
		I	TDI: Test Data para a interface *JTAG*. Este pino foi utilizado com este propósito para *JTAG* externo e também é compartilhado com o *FPGA*.
64	*P1.27/TDO*	I/O	P1.27: Pino de propósito geral, entrada e saída digital.
		O	*TDO*: *Test Data Output* para a interface *JTAG*. Este pino foi utilizado com este propósito para *JTAG* externo e também é compartilhado com o *FPGA*.
24	*P1.26/RTCK*	I/O	P1.26: Pino de propósito geral, entrada e saída digital.
			RTCK: Returned Test Clock output. Sinal extra adicionado ao *JTAG port*. Auxilia processo de sincronização de debugger quando a frequência do processador varia. Pino bidirecional com *pull-up* interno. Nota: Nível baixo neste pino durante o *Reset*, força o nível dos pinos P1.31:26 para operarem como *debug port* após *reset*. Este pino foi utilizado com este propósito para *JTAG* externo.
28	*P1.25/EXTIN0*	I/O	P1.25: Pino de propósito geral, entrada e saída digital. Este pino está sendo utilizado no Botão BT6.
		I	*EXTIN0: External Trigger Input*.
32	*P1.24/TRACECLK*	I/O	P1.24: Pino de propósito geral, entrada e saída digital. Utilizado como *JTRST* para a interface *JTAG* externo.
		O	*TRACECLK*: *Trace Clock*. Padrão I/O com *pull-up* interno.
36	*P1.23/PIPESTAT2*	I/O	P1.23: Pino de propósito geral, entrada e saída digital. Este pino está sendo utilizado no Botão BT5.
			PIPESTAT2: *Pipeline Status,* bit 2. Padrão I/O com *pull-up* interno.
40	*P1.22/PIPESTAT1*	I/O	P1.22: Pino de propósito geral, entrada e saída digital. Este pino está sendo utilizado no Botão BT4.
			Pipeline Status, bit 1. Padrão I/O com *pull-up* interno.
44	*P1.21/ PIPESTAT0*	I/O	P1.21: Pino de propósito geral, entrada e saída digital. Este pino está sendo utilizado no Botão BT3.
			PIPESTAT0: Pipeline Status, bit 0. Padrão I/O com *pull-up* interno.
48	*P1.20/ TRACESYNC*	I/O	P1.20: Pino de propósito geral, entrada e saída digital. Este pino está sendo utilizado no LED D6.

		O	*TRACESYNC: Trace Synchronization.* Padrão I/O com *pull-up* interno. Nota: Nível baixo neste pino durante o *Reset*, reduz o nível dos pinos P1.25:16 para operarem como *TRACE port* após *reset*.
4	*P1.19/ TRACEPKT3*	I/O	P1.19: Pino de propósito geral, entrada e saída digital. Este pino está sendo utilizado no LED D7.
		O	*TRACEPKT3*: *Trace Packet*, bit 3. Padrão I/O com *pull-up* interno.
8	*P1.18/ TRACEPKT2*	I/O	P1.18: Pino de propósito geral, entrada e saída digital. Este pino está sendo utilizado no LED D8.
		O	*TRACEPKT2: Trace Packet,* bit 3. Padrão I/O com *pull-up* interno.
12	*P1.17/ TRACEPKT1*	I/O	P1.17: Pino de propósito geral, entrada e saída digital. Este pino está sendo utilizado no LED D9.
		O	*TRACEPKT1: Trace Packet,* bit 3. Padrão I/O com *pull-up* interno.
16	*P1.16/ TRACEPKT0*	I/O	P1.16: Pino de propósito geral, entrada e saída digital. Este pino está sendo utilizado no LED D10.
		O	*TRACEPKT0: Trace Packet,* bit 3. Padrão I/O com *pull-up* interno.
17	*P0.31/UP_LED/ CONNECT*	O	P0.31: Pino de propósito geral, saída digital – *General Purpose Output - GPO.*
		O	*UP_LED: LED* indicador de bom *link USB*. Este pino está sendo utilizado para esta função.
		O	*CONNECT:* Sinal utilizado para chavear resistor externo de 1.5 kΩ sob controle de *software* (a ativação deste sinal é em *LOW*).
15	*P0.30/AD0.3/ EINT3/CAP0.0*	I/O	P0.30: Pino de propósito geral, entrada e saída digital. Este pino está sendo utilizado no Botão BT2.
		I	AD0.3: Conversor *A/D* 0, entrada 3. Esta entrada analógica está sempre conectada a este pino.
		I	*EINT3:* Entrada externa de interrupção 3.
		I	*CAP0.0:* Entrada *Capture* para *Timer 0*, canal 0.
14	*P0.29/AD0.2/ CAP0.3/MAT0.3*	I/O	P0.29: Pino de propósito geral, entrada e saída digital. Este pino está sendo utilizado como uma das saídas da interface *SPI.*
		I	*AD0.2:* Conversor *A/D* 0, entrada 2. Esta entrada analógica está sempre conectada a este pino.
		I	*CAP0.3:* Entrada *Capture* para *Timer 0*, canal 3.
		O	*MAT0.3*: Saída *Match* para *Timer 0*, canal 3.
13	*P0.28/AD0.1/ CAP0.2/MAT0.2*	I/O	P0.28: Pino de propósito geral, entrada e saída digital. Este pino está sendo utilizado como uma das saídas da interface *SPI.*
		I	AD0.1: Conversor *A/D* 0, entrada 1. Esta entrada analógica está sempre conectada a este pino.
		I	*CAP0.2:* Entrada *Capture* para *Timer 0*, canal 2.
		O	*MAT0.2:* Saída *Match* para *Timer 0*, canal 2.
9	*P0.25/AD0.4/Aout*	I/O	P0.25: Pino de propósito geral, entrada e saída digital.

		I	AD0.4: Conversor *A/D* 0, entrada 4. Esta entrada analógica está sempre conectada a este pino. Este pino está sendo utilizado no potenciômetro.
		O	*Aout:* Saída do conversor *A/D*.
58	*P0.23/Vbus*	I/O	P0.23: Pino de propósito geral, entrada e saída digital.
		I	*Vbus:* Indica a presença de alimentação *USB*. Este pino está sendo utilizado para esta função.
2	*P0.22/AD1.7/ CAP0.0/MAT0.0*	I/O	P0.22: Pino de propósito geral, entrada e saída digital. Este pino está sendo utilizado como uma das saídas da interface *SPI*.
		I	*AD1.7:* Conversor *A/D* 1, entrada 7. Esta entrada analógica está sempre conectada a este pino.
		I	*CAP0.0:* Entrada *Capture* para *Timer 0*, canal 0.
		O	*MAT0.0*: Saída *Match* para *Timer 0*, canal 0.
1	*P0.21/PWM5/ AD1.6/CAP1.3*	I/O	P0.21: Pino de propósito geral, entrada e saída digital. Este pino está sendo utilizado como uma das saídas da interface *SPI.*
		O	*PWM5*: *Pulse width modulator* saída 5.
		I	*AD1.6*: Conversor *A/D* 1, entrada 6. Esta entrada analógica está sempre conectada a este pino.
		I	*CAP1.3*: Entrada *Capture* para *Timer 1*, canal 3.
55	*P0.20/ MAT1.3/ SSEL1/EINT3*	I/O	P0.20: Pino de propósito geral, entrada e saída digital.
		O	*MAT1.3*: Saída *Match* para *Timer 1*, canal 3.
		I	SSEL1: *Slave Select* para *SSP*. Configura a interface *SSP* como *slave*. Este pino é utilizado com esta função na comunicação *SPI* com o *FPGA*.
		I	EINT3: Entrada externa de interrupção 3.
54	*P0.19/MAT1.2/ MOSI1/CAP1.2*	I/O	P0.19: Pino de propósito geral, entrada e saída digital.
		O	*MAT1.2:* Saída *Match* para *Timer 1*, canal 2.
		I/O	*MOSI1: Master Out Slave In* para *SSP*. Saída de dados da *SSP master* ou entrada de dados para *SSP slave*. Este pino é utilizado com esta função na comunicação *SPI* com o *FPGA*.
		I	*CAP1.2*: Entrada *Capture* para *Timer 1*, canal 2.
53	*P0.18/CAP1.3/ MISO1/MAT1.3*	I/O	P0.18: Pino de propósito geral, entrada e saída digital.
		I	*CAP1.2:* Entrada *Capture* para *Timer 1*, canal 3.
		I/O	*MISO1*: *Master In Slave Out* para *SSP*. Entrada de dados para *SPI master* ou saída de dados para *SSP slave.* Este pino é utilizado com esta função na comunicação *SPI* com o *FPGA*.
		O	*MAT1.3*: Saída *Match* para *Timer 1*, canal 3.
47	*P0.17/CAP1.2/ SCK1/MAT1.2*	I/O	P0.17: Pino de propósito geral, entrada e saída digital.
		I	*CAP1.2*: Entrada *Capture* para *Timer 1*, canal 2.
		I/O	*SCK1*: *Serial Clock* para *SSP*. Saída de *clock* para *máster* ou entrada para *slave*. Este pino é utilizado com esta função na comunicação SPI com o *FPGA*.
		O	*MAT1.2*: Saída *Match* para *Timer 1*, canal 2.
46	*P0.16/EINT0/*	I/O	P0.16: Pino de propósito geral, entrada e saída digital.

	MAT0.2/CAP0.2	I	*EINT0:* Entrada externa de interrupção 0. Este pino foi ligado direto ao *FPGA* de modo que ele possa interromper a execução do *ARM*.
		O	*MAT0.2:* Saída *Match* para *Timer 0*, canal 2.
		I	*CAP0.2*: Entrada *Capture* para *Timer 0*, canal 2.
45	*P0.15/RI1/ EINT2/AD1.5*	I/O	P0.15: Pino de propósito geral, entrada e saída digital.
		I	*RI1:* Entrada *ring indicator* da *UART1*.
		I	*EINT2*: Entrada externa de interrupção 0. Este pino foi ligado direto ao *FPGA* de modo que ele possa interromper a execução do *ARM*.
		I	*AD1.5:* Conversor *A/D* 1, entrada 5. Esta entrada analógica está sempre conectada a este pino.
41	*P0.14/DCD1/ EINT1/SDA1*	I/O	P0.14: Pino de propósito geral, entrada e saída digital.
		I	*DCD1*: Entrada *Data Carrier Input* da *UART1*.
		I	*EINT1*: Entrada externa de interrupção 1.
		I/O	*SDA1: I^2C1 data input/output*. Saída de coletor aberto. Nota: Nível baixo neste pino enquanto *reset* faz com que o *boot-loader* embarcado (*on-chip*) assuma o controle após a inicialização. Este pino está sendo compartilhado entre o controlador *ISP/IAP* e *SDA1 (I^2C)* ligado ao *FPGA* por meio de um seletor P3.
39	*P0.13/DTR1/ MAT1.1/AD1.4*	I/O	P0.13: Pino de propósito geral, entrada e saída digital. Este pino é utilizado com esta função na comunicação SPI com o *FPGA*.
		O	*DTR1:* Saída *data terminal ready* para *UART1*.
		O	*MAT1.1*: Saída *match* para *Timer 1*, canal 1.
		I	*AD1.4*: Conversor *A/D* 1, entrada 4. Esta entrada analógica está sempre conectada a este pino.
38	*P0.12/DSR1/ MAT1.0/AD1.3*	I/O	P0.12: Pino de propósito geral, entrada e saída digital. Este pino é utilizado com esta função na comunicação *SPI* com o *FPGA*.
		I	*DSR1:* Entrada *data set ready* para *UART1*.
		O	*MAT1.0*: Saída *match* para *Timer 1*, canal 0.
		I	*AD1.3:* Conversor *A/D* 1, entrada 3. Esta entrada analógica está sempre conectada a este pino.
37	*P0.11/CTS1/ CAP1.1/SCL1*	I/O	P0.12: Pino de propósito geral, entrada e saída digital.
		I	*CTS1*: Entrada *clear to send* para *UART1*.
		I	*CAP1.1:* Entrada *capture* para *timer 1*, canal 1.
		I/O	*SCL1*: entrada e saída de *clock, I^2C1*. Saída dreno aberto. Este pino é utilizado com esta função na comunicação I^2C com o *FPGA*.
35	*P0.10/RTS1/ CAP1.0/AD1.2*	I/O	P0.10: Pino de propósito geral, entrada e saída digital.
		O	*RTS1*: Saída *request to send* para *UART1*. Este pino é utilizado com esta função na comunicação *SPI* com o *FPGA*.
		I	*CAP1.0*: Entrada *Capture* para *Timer 1*, canal 0.
		I	*AD1.1:* Conversor *A/D* 1, entrada 1. Esta entrada analógica está sempre conectada a este pino.
34	*P0.9/RxD1/*	I/O	P0.9: Pino de propósito geral, entrada e saída digital.

	PWM6/EINT3	I	*RxD1:* Entrada *receiver* para *UART1*. Este pino é utilizado com esta função na comunicação externa *RS232.* Ele também é compartilhado com o *FPGA* por meio de um seletor (JP7), de acordo com a necessidade do usuário.
		O	*PWM6:* Saída 6 com modulação por largura de pulso (*Pulse Width Modulator*).
		I	*EINT3:* Entrada externa de interrupção, 3.
33	*P0.8/TXD1/ PWM4/AD1.1*	I/O	P0.8: Pino de propósito geral, entrada e saída digital.
		O	*TxD1:* Saída *transmitter* para *UART1*. Este pino é utilizado com esta função na comunicação externa *RS232.* Ele também é compartilhado com o *FPGA* por meio de um seletor (JP6), de acordo com a necessidade do usuário.
		O	*PWM4:* Saída *pulse width modulator*, 4.
		I	*AD1.1:* Conversor *A/D* 1, entrada 1. Esta entrada analógica está sempre conectada a este pino.
31	*P0.7/SSEL0/ PWM2/EINT2*	I/O	P0.7: Pino de propósito geral, entrada e saída digital.
		I	*SSEL0:* Entrada Slave Select para *SPI0*. Seleciona a interface *SPI* como *slave*. Este pino é utilizado com esta função na comunicação *SPI* externa.
		O	*PWM2:* Saída 2 com modulação por largura de pulso (*pulse width modulator*).
		I	*EINT2:* Entrada externa de interrupção, 2.
30	*P0.6/MOSI0/ CAP0.2/AD1.0*	I/O	P0.6: Pino de propósito geral, entrada e saída digital.
		I/O	*MOSI0: Master Out Slave In* para SSP. Saída de dados da *SSP master* ou entrada de dados para *SSP slave*. Este pino é utilizado com esta função na comunicação externa SPI.
		I	*CAP0.2:* Entrada *capture* para *timer 0*, canal 2.
		I	*AD1.0:* Conversor *A/D* 1, entrada 0. Esta entrada analógica está sempre conectada a este pino.
29	*P0.5/MISO0/ MAT0.1/AD0.7*	I/O	P0.5: Pino de propósito geral, entrada e saída digital.
		I/O	*MISO0*: *Master In Slave Out* para *SSP*. Entrada de dados para SPI *master* ou saída de dados para SSP *slave*. Este pino é utilizado com esta função na comunicação externa *SPI*.
		O	*MAT1.3*: Saída *match* para *timer 0*, canal 1.
		I	*AD0.7*: Conversor *A/D* 0, entrada 7. Esta entrada analógica está sempre conectada a este pino.
27	*P0.4/SCK0/ CAP0.1/AD0.6*	I/O	P0.4: Pino de propósito geral, entrada e saída digital.
		I/O	*SCK0*: *Serial Clock* para *SSP*. Saída de *clock* para *master* ou entrada para *slave*. Este pino é utilizado com esta função na comunicação externa SPI.
		I	*CAP0.1*: Entrada *capture* para *timer 0*, canal 1.
		I	*AD0.6:* Conversor *A/D* 0, entrada 6. Esta entrada analógica está sempre conectada a este pino.
26	*P0.3/SDA0/*	I/O	P0.3: Pino de propósito geral, entrada e saída digital.

	MAT0.0/EINT1	I/O	*SDA0: I^2C0* data *input/output.* Saída de coletor aberto. Este pino está sendo compartilhado entre o *SDA0* externo (*I^2C*) quanto para a memória *EEPROM* externa por meio do seletor P2.
		O	*MAT0.0*: Saída *match* para *timer 0*, canal 0.
		I	*EINT1*: Entrada externa de interrupção, 1.
22	*P0.2/SCL0/ CAP0.0*	I/O	P0.2: Pino de propósito geral, entrada e saída digital.
		I/O	*SCL0*: entrada e saída de *clock*, *I^2C1*. Saída dreno aberto. Este pino é compartilhada com esta função na comunicação externa *I^2C* e memória *EEPROM* externa por meio do seletor P1.
		I	CAP0.0: Entrada *capture* para *timer 0*, canal 0.
21	*P0.1/RxD0/ PWM3/EINT0*	I/O	P0.1: Pino de propósito geral, entrada e saída digital.
		I	*RxD0*: Entrada *receiver* para *UART0*. Este pino é utilizado com esta função na comunicação externa *RS232*. Ele também é compartilhado com o *FPGA* por meio de um seletor (JP5), de acordo com a necessidade do usuário.
		O	*PWM3*: Saída 3 com modulação por largura de pulso (*pulse width modulator*).
		I	*EINT0*: Entrada externa de interrupção, 0.
19	*P0.0/TXD0/ PWM1*	I/O	P0.0: Pino de propósito geral, entrada e saída digital.
		O	*TxD0*: Saída *transmitter* para *UART0*. Este pino é utilizado com esta função na comunicação externa *RS232*. Ele também é compartilhado com o *FPGA* por meio de um seletor (JP4), de acordo com a vontade do usuário.
		O	PWM1: Saída *pulse width modulator*, 1.

4.3.6 PROJETO DO MODULO DE ALIMENTAÇÃO DO FPGA (FPGA-ALIMENTAÇÃO)

Após feita toda a montagem do circuito referente ao *ARM*, partiu-se para a implementação dos circuitos que dão suporte ao *FPGA*.

O subsistema *FPGA-Alimentação* agrega toda a parte de alimentação deste dispositivo. Este módulo constitui-se de dois circuitos: (a) regulador de tensão e (b) distribuidor de *clock*.

4.3.6.1 PROJETO DO SISTEMA DE ALIMENTAÇÃO PARA O FPGA (3V3, 2V5 E 1V2)

Diferentemente do *ARM*, o *FPGA* possui múltiplas entradas de tensão a serem utilizadas simultaneamente. São duas entradas de alimentação para funções logicas internas, *Vccint* e *Vccaux*. Cada um dos quatro bancos de entrada e cada saída de sinais possui sua própria entrada *Vcco*, que alimenta a saída do *buffer* associado ao respectivo banco. Dessa forma, todas as conexões de cada banco devem ser feitas utilizando a mesma voltagem. A Tabela 15 apresenta o resumo das tensões utilizadas.

Tabela 15 – Resumo das voltagens de alimentação da Spartan-3E - *FPGA*.

Entrada de alimentação	***Descrição***	***Voltagem nominal***
Vccint	Voltagem de alimentação do núcleo interno. Alimenta todas as funções lógicas internas como, *CLBs*, bloco de *RAM* e multiplicadores. Entrada do circuito *Power-On Reset (POR)*.	1,2V
Vccaux	Voltagem de alimentação auxiliar. Alimenta os gerenciadores de *clock* digitais (DCMs), *drivers* diferenciais, pinos de configuração dedicados, interface *JTAG*. Entrada do circuito *Power-On Reset* (*POR*).	2,5V
Vcco_0	Alimenta a saída dos *buffers* das entradas e saídas do Banco 0. O Banco 0 situa-se na borda superior do *FPGA*.	Livre escolha: 3,3V, 2,5V ou 1,2V.
Vcco_1	Alimenta a saída dos *buffers* das entradas e saídas do Banco 1. O Banco 1 situa-se na borda direita do *FPGA*. Quando em modo *flash* paralelo, utilizando interface *BPI* (*Byte-Wide Peripheral Interface)*, deve ser conectado com a mesma voltagem que a *Flash PROM*.	Livre escolha: 3,3V, 2,5V ou 1,2V.
Vcco_2	Alimenta a saída dos buffers das entradas e saídas do Banco 2. O Banco 2 situa-se na borda inferior do *FPGA*. Conectar com a mesma fonte de voltagem que a fonte de configuração do *FPGA*. Entrada do circuito *Power-On Reset (POR).*	Livre escolha: 3,3V, 2,5V ou 1,2V.
Vcco_3	Alimenta a saída dos *buffers* das entradas e saídas do Banco 3. O Banco 3 situa-se na borda superior do *FPGA*.	Livre escolha: 3,3V, 2,5V ou 1,2V.

Em aplicações que utilizam apenas fonte 3V3, todos os quatro bancos devem ser alimentados por esta fonte. Entretanto, os *FPGAs* Spartan-3E proporcionam a possibilidade de intercambiar entre diferentes voltagens de entrada e saída para *Vcco* de diferentes bancos. Cada banco *I/O* também possui uma entrada de tensão de referência, chamada *Vref*. Se o banco *I/O* inclui entradas e saídas padrão que requerem uma voltagem de referência então todos os pinos *Vref*, do referido banco, devem ser conectados na mesma tensão.

Vários fabricantes de fontes de alimentação oferecem soluções completas voltadas aos *FPGAs Xilinx*. Algumas soluções já integram as 3 principais tensões em um único circuito integrado. Dessa forma, optou-se por utilizar uma dessas soluções, pois são projetadas especificamente para o *FPGA* utilizado e sugeridas pelo próprio fabricante.

Para este projeto, optou-se por utilizar o CI *TPS75003* do fabricante *Texas Instruments* desenvolvido e testado para suprir *FPGAs Spartan-3E*. A partir disso, seguiu-se às recomendações e sugestões da documentação do próprio CI. Além do regulador fixo de tensão de 1.2V/3 Amperes, o CI ainda disponibiliza duas outras fontes ajustáveis. Optou-se pela configuração de 2V5 e 3V3. A tensão 2V5 é recomendável para a interface *JTAG*, e o 3V3 é pelo fato de o *FPGA* comunicar-se com o *ARM*, que é alimentado com esta tensão também. A Figura 17 mostra o esquemático do circuito regulador de tensão implementado, conforme o manual do fabricante do *TPS75003*.

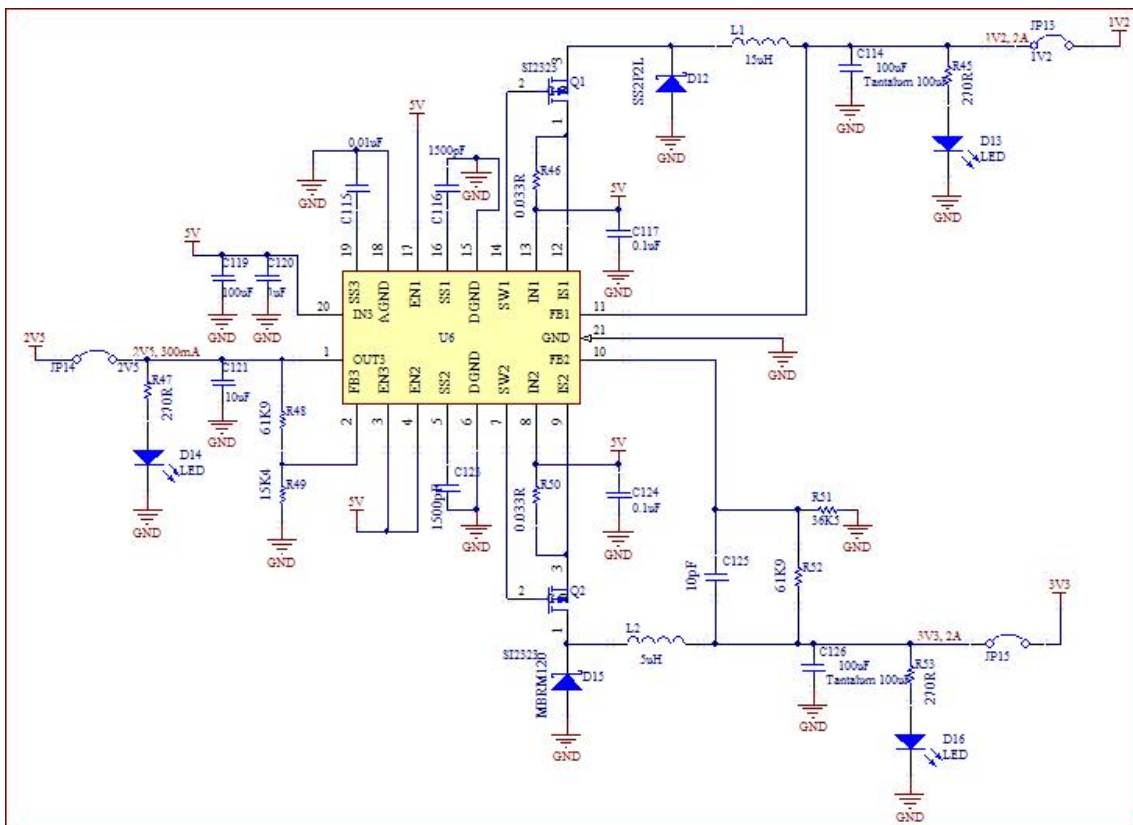



Figura 17 – Circuito regulador de tensão – 1V2, 2V5 e 3V3 - *FPGA*.

A fim de dar maior versatilidade à placa foram instalados *jumpers* na saída de cada tensão. Com isso, é possível testar o funcionamento do sistema de alimentação da placa em cada saída de maneira independente.

Ao fazer uso de cada valor de alimentação, é preciso ter atenção à corrente que será exigida em cada projeto. Os valores nominais de tensão/corrente para cada saída são respectivamente: 1V2/2A, 2V5/300mA, 3V3/2A.

4.3.6.2 PROJETO DO DISTRIBUIDOR DE CLOCK PARA O FPGA

Sistemas modernos normalmente exigem interfaces de troca de dados síncronas assim como sistemas analógicos de radio frequência que exigem um tempo de referência comum, para garantir assim, um amplo desempenho e segurança.

Tendo isso como referência inseriu-se um circuito distribuidor de *clock* que possibilita o desenvolvimento de sistemas síncronos sem utilizar dispositivos externos para isso. O oscilador utilizado foi de 50MHz, sendo que os sinais distribuídos foram conduzidos aos *global-clocks* (GCLK) dos bancos 0 e 2, por opção de projeto. A Figura 18 mostra esquematicamente o circuito utilizado para distribuição de *clock*.

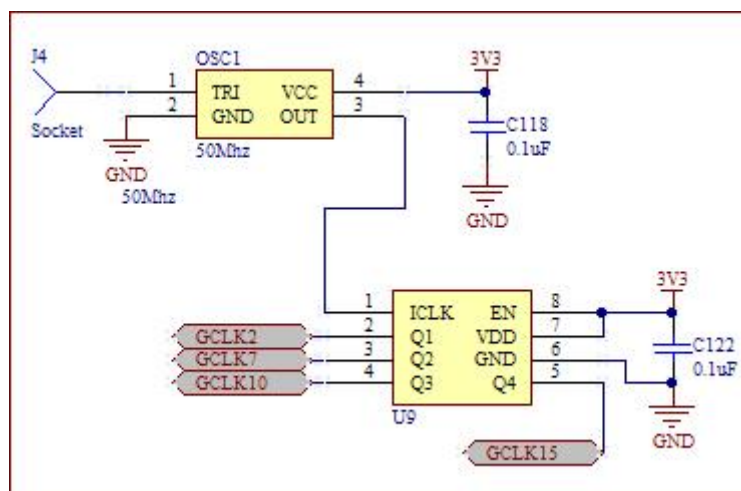



Figura 18 – Circuito distribuidor de *clock* – *FPGA*.

4.3.7 MÓDULO DE MEMÓRIA PARA O FPGA (FPGA - MEMÓRIA)

O próprio *FPGA* já traz em si 73k *Bits* de memória *RAM* distribuída, e 360K de *RAM* em bloco. De modo a viabilizar projetos mais robustos, especialmente voltados à aquisição, análise, monitoramento e controle de dados, optou-se por inserir uma memória externa ao *FPGA*. Foi selecionada uma *EEPROM Flash* com capacidade de armazenamento de 4 *Mega bytes*. Foi escolhido o CI *XCF04SV* do fabricante *Xilinx Inc.*, cujo tipo de programação utilizada é o ISP

(*in-system programmable*). Este CI é recomendado pela *Xilinx Inc.* pela compatibilidade com o *FPGA* utilizado (Xilinx, 2008). O circuito implementado foi extraído de uma nota de aplicação da família *Spartan-3*[21] e, nesta nota, o circuito sugerido já integra a comunicação externa *JTAG* à topologia a ser utilizada. A Figura 19 ilustra a montagem implementada na placa de circuito impresso. Nessa configuração o *FPGA* e a memória *flash* são ligados em série ao computador, facilitando assim a programação de ambos.

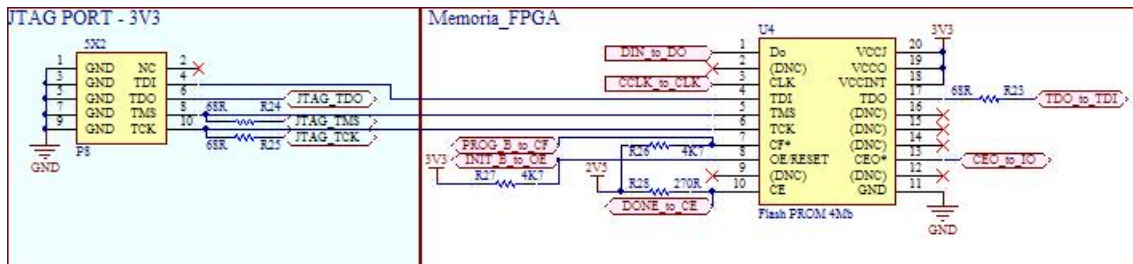



Figura 19 – Circuito implementado para memória externa do *FPGA* utilizando interface *JTAG*, na versão V1.1.

Pode ser observado que o conector P8 é a interface de saída da comunicação *JTAG*, sendo que a equivalência de pinos pode ser encontrada na Tabela 16.

Tabela 16 – Descrição dos pinos do conector P8, *JTAG* externo do *FPGA*.

PINO	*NOME*	*DESCRIÇÃO*
1	GND	*Ground*
2	NC	*Not Connected*
3	GND	*Ground*
4	TDI	*Test Data In*
5	GND	*Ground*
6	TDO	*Test Data Out*
7	GND	*Ground*
8	TMS	*Test Mode Select*
9	GND	*Ground*
10	TCK	*Test Clock*

Ao se fazer os primeiros testes de gravação, ainda no *iMPACT* (maiores detalhes no CAPÍTULO 5), percebeu-se que o pacote *ISE* não reconhecia corretamente o *FPGA* e tampouco a memória associada a ele. O circuito implementado possibilita a configuração *master-serial* do *FPGA*, associado a uma memória *PROM*, segundo o modelo de referência da Figura 20. Esse modelo está implementado na plataforma desenvolvida (versão 1.1).

[21] *Application Note: The 3.3V Configuration of Spartan-3 FPGAs.* Disponível em: http://www.xilinx.com/support/documentation/application_notes/xapp453.pdf Acesso em 01/12/2010.

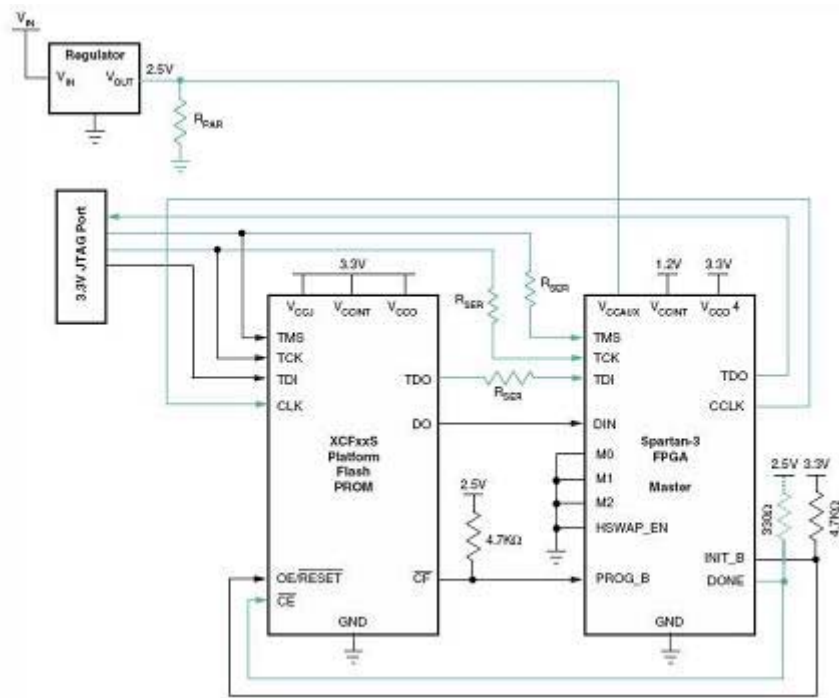



Figura 20 – Configuração utilizada na versão 1.1 para comunicação *FPGA*/FLASH [22].

Ao não se obter sucesso na gravação do circuito nessa configuração, optou-se então por adaptar a plataforma a uma nova configuração, disponível documentação do *FPGA* utilizado, conforme mostrada na Figura 21.

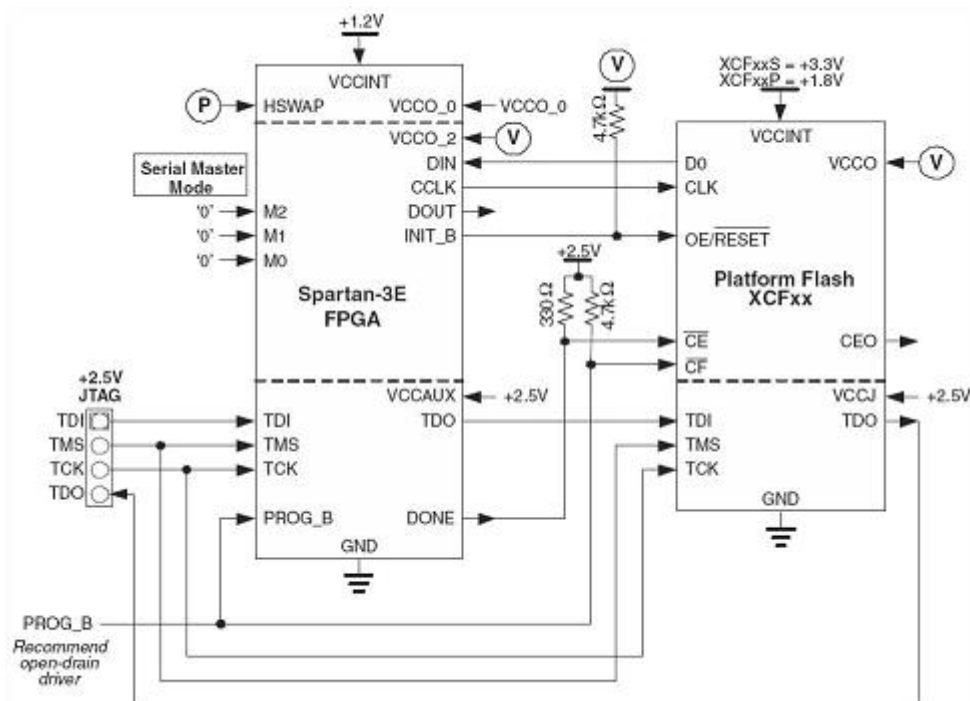
---

[22] Disponível em: http://www.xilinx.com/support/documentation/application_notes/xapp453.pdf. Acesso em: 27/06/2010.

Figura 21 – Nova configuração proposta para comunicação *FPGA*/FLASH[23].

Para se conseguir esse novo esquemático foi preciso alterar fisicamente algumas ligações e trilhas da plataforma por meio de vias, interrupção de trilhas e soldagem de fios condutores. Isso porque o problema foi detectado somente após a confecção da placa, na fase de testes. Após a montagem da nova configuração o pacote de programação reconheceu adequadamente o *FPGA* e a Memória PROM. A nova configuração está proposta e devidamente documentada na sessão *SUGESTÕES PARA TRABALHOS FUTUROS* no CAPÍTULO 6.

4.3.8 PROJETO DE INTERFACES PARA O FPGA

Assim como no *ARM*, também foi criado um módulo de interfaces com o usuário. Este módulo congrega circuitos dedicados à interação com o usuário ou desenvolvedor. Visando atender as demandas de projetos que dependam de entrada e retorno do próprio usuário foram implementadas interfaces com: (a) botões, (b) potenciômetro e (c) *LEDs*.

4.3.8.1 INTERFACE DE ENTRADAS DIGITAIS (FPGA-PUSH BUTTON)

Foram elaborados dois módulos para entrada de dados: (a) botões e (b) potenciômetro. No primeiro caso, optou-se por utilizar oito botões de interface. Esses botões podem ser utilizados independentemente ou em conjunto conforme a necessidade de entrada de parâmetros. A Figura 22 mostra a configuração utilizada.

---

[23] Disponível em: http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf. Acesso em: 27/06/2012.

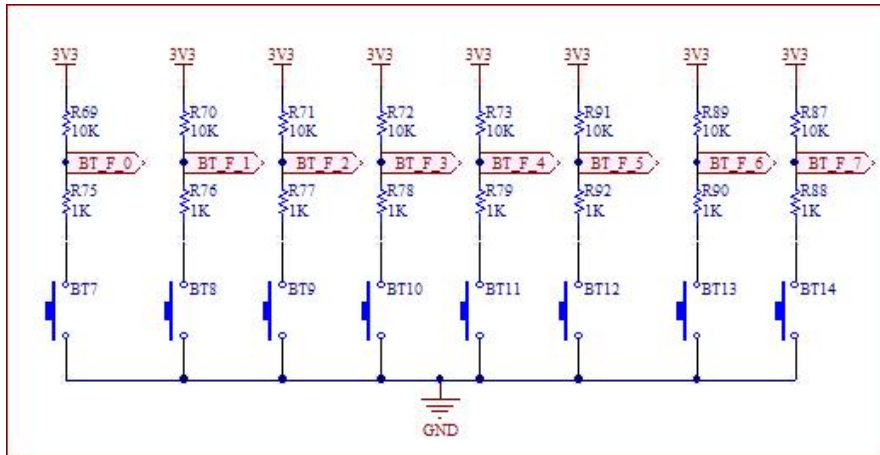



Figura 22 – Circuito de interface de entrada com botões – *FPGA-Push button*.

A nomenclatura utilizada para cada botão, BT_F_X, corresponde a *botão*, em inglês *push-button,* ligado ao *FPGA* seguido de seu respectivo numero sequencial. Cada botão está ligado a um pino correspondente no *FPGA*, conforme a Tabela 17.

Tabela 17 – Correspondência entre botões de interface e pinos no *FPGA*.

SEQUÊNCIA	*BOTÃO*	*PINO NO FPGA*
1	BT7	77
2	BT8	83
3	BT9	84
4	BT10	86
5	BT11	87
6	BT12	69
7	BT13	74
8	BT14	98

4.3.8.2 INTERFACE DE ENTRADA ANALÓGICA (FPGA-POTENCIÔMETRO)

Para o segundo caso, foi inserido um potenciômetro de 10KΩ, de modo a dar mais versatilidade à placa com uma entrada variável, multivalorada. A Figura 23 mostra o circuito utilizado para inserir o potenciômetro. A nomenclatura *POT_F* corresponde a potenciômetro ligado ao *FPGA*.

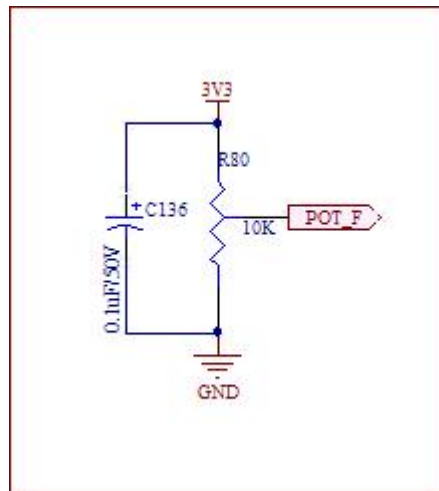



Figura 23 – Circuito do potenciômetro ligado ao *FPGA (FPGA-Potenciômetro)*.

4.3.8.3 INTERFACES DE SAÍDAS DIGITAIS DO FPGA (FPGA-LEDs)

Além das interfaces de entrada, foi implementada ainda uma interface de saída composta por oito *LEDs*. Interfaces de saída são importantes especialmente durante a fase de desenvolvimento de novos projetos, pois elas servem como retorno visual em testes de funcionamento de partes de algoritmos. Além disso, pode servir também para notificação, alerta ou sinalização ao usuário. A Figura 24 mostra o circuito utilizado para a interface com *LEDs*.

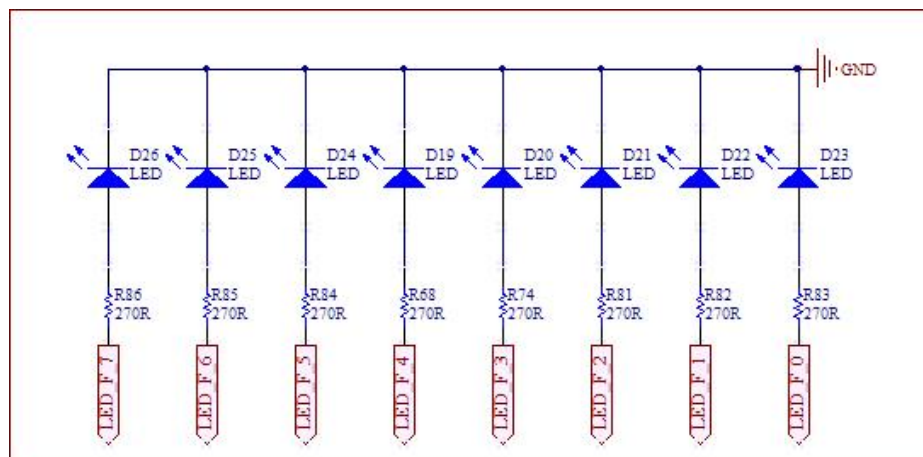



Figura 24 – Circuito de interface de saída *LEDs* - *FPGA*.

A nomenclatura utilizada para cada LED, *LED_F_X*, corresponde a *LED* ligado ao *FPGA* seguido de seu respectivo numero sequencial. Cada *LED* está ligado a um pino correspondente no *FPGA*, conforme a Tabela 18.

Tabela 18 – Correspondência entre *LEDs* de interface e pinos no *FPGA*.

SEQUÊNCIA	***LED***	***PINO NO ARM***
1	D23	89
2	D22	93
3	D21	94
4	D20	96
5	D19	97
6	D24	99
7	D25	100
8	D26	102

4.3.9 PROJETO DO CIRCUITO ETHERNET PARA O FPGA

A partir da proposta inicial de se desenvolver uma plataforma que atendesse também aplicações em computação ubíqua que, por natureza, exigem interação e interoperabilidade com outros dispositivos, incluir comunicação *Ethernet* no sistema tornou-se essencial.

A rede *Ethernet* é padronizada pela norma *IEEE 802.3*. Ela é do tipo barramento e constitui uma das opções para a camada *host/rede* do modelo de referência *TCP/IP*. Cada dispositivo *Ethernet* deve possuir um número *MAC* (*Media Access Control*) que o identifica unicamente. Ele é formado por um conjunto de seis números hexadecimais de dois dígitos (48 bits). Todos os dispositivos se interligam por um *HUB*, que nada mais é que um concentrador, recebendo dados de um dispositivo e repassando aos demais.

A comunicação em uma estrutura desse tipo exige que camadas superiores peçam o envio de um determinado pacote a um dispositivo com certo número *IP*. Assim, deve-se procurar na rede qual dispositivo possui aquele *IP*, para então se obter o seu endereço *MAC*. É através desse endereço que um dispositivo consegue se comunicar com aquele desejado através do meio físico (*Ethernet*). Isso é feito através de um protocolo chamado *ARP* (*Address Resolution Protocol*). Esse protocolo envia sinais *broadcast* (para todos na rede) *perguntando* qual dispositivo possui aquele endereço *IP*. Então o dispositivo retorna uma mensagem que contém o seu endereço *MAC*. A partir daí, a camada de *host*/*network* poderá enviar a informação pelo meio físico até o dispositivo de destino.

Outro serviço notório é o *DHCP* (*Dynamic Host Configuration Protocol*), que é utilizado para a configuração automática dos dispositivos em uma rede *IP* de tal forma que o endereço *IP* e outras informações sejam fornecidas assim que o dispositivo é ligado. Para isso, o dispositivo envia um pacote *UDP/IP* em modo *broadcast* através de determinada porta. O equipamento responsável por atender a esse pedido é o servidor de *DHCP*. Este dispositivo lê esse pacote,

decidindo qual *IP* será assimilado por esse novo dispositivo e envia essas informações através da mesma porta *UDP* até o cliente que realizou a requisição. A partir desse momento, o dispositivo adota que o seu endereço *IP* é aquele retornado pelo *DHCP server*.

Definida a necessidade do protocolo *Ethernet*, partiu-se para a escolha do dispositivo controlador do protocolo. Para valorizar trabalhos anteriormente desenvolvidos dentro do departamento de engenharia mecatrônica da UnB, escolheu-se utilizar o controlador *ENC28J60* que já havia sido utilizado em trabalhos anteriores, testado e aprovado por colegas.

Esse controlador desenvolvido pela empresa *Microchip* é responsável por construir uma interface *serial/Ethernet*. Ele deve ser utilizado em conjunto com outro microcontrolador/*FPGA* com suporte ao protocolo *SPI*. A Figura 25 mostra o esquemático desse controlador. Pode-se observar que, internamente, está construído a camada *MAC* e a camada física (*PHY*) da *Ethernet*. Logo, os dados que devem ser enviados para ele é o pacote *IP*.

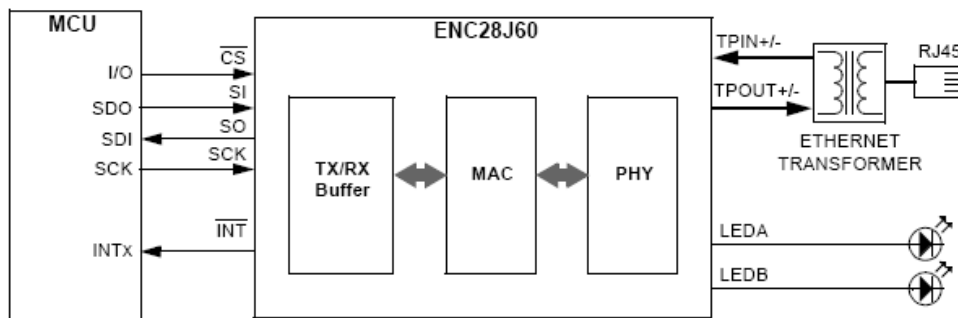



Figura 25 - Esquemático do controlador Ethernet.

As principais características deste controlador são:

a) Compatível com o padrão *IEEE 802.3 (Ethernet);*

b) Compatível com as redes 10/100/1000 Base-T;

c) Suporta uma porta 10 Base-T com detecção automática de polaridade de correção;

d) Suporta comunicação *full* e *half-duplex*;

e) Pode ser programado para retransmissão automática na colisão[24];

f) Verifica automaticamente pacotes errados e os rejeita;

g) 8 KB para o *buffer* de transmissão e recepção;

[24] Uma colisão é quando dois dispositivos começam a transmitir em uma rede Ethernet. Nesse caso, os dois dispositivos devem parar a transmissão, esperar um determinado tempo e tentar reiniciá-la. Esse é o motivo pelo qual a rede Ethernet não é determinística.

h) Interface *SPI* com *clock* de até 20MHz;

i) Alimentação de 3.1 à 3.6V (3.3V típico);

j) Entradas tolerantes a 5V;

k) Exige *clock* externo de 25MHz.

A pilha *TCP/IP* desenvolvida pela *Microchip* já fornece toda a interface necessária. Basicamente, o funcionamento do controlador se dá escrevendo nos registradores e na memória interna do *ENC28J60* através de comandos enviados pela porta *SPI*. Logo, o controlador envia os dados para um dispositivo, configura-se o *MAC* de destino e em seguida é colocado no *buffer* o pacote desejado para a transmissão. Após essa configuração, um comando é enviado para que se inicie a transmissão.

A recepção ocorre de maneira análoga, o pacote recebido é colocado no *buffer* de transmissão até que o programa o leia. O *ENC28J60* avisa ao microcontrolador/*FPGA* da existência de um pacote através de interrupção ou através de *pooling*. O primeiro método interrompe a execução do programa no microcontrolador/*FPGA* e passa para a rotina programada que será encarregada de pegar o pacote recebido. No segundo método, o microcontrolador/*FPGA* de forma periódica *pergunta* ao *ENC28J60* se existe algum dado no *buffer* e, caso exista, faz a transferência.

O circuito utilizado foi extraído do manual do fabricante do controlador que sugere tanto o circuito quanto os valores dos dispositivos. Este circuito também foi utilizado por colegas do Departamento em trabalhos anteriores (Chagas, 2008). A Figura 26 mostra o circuito implementado.

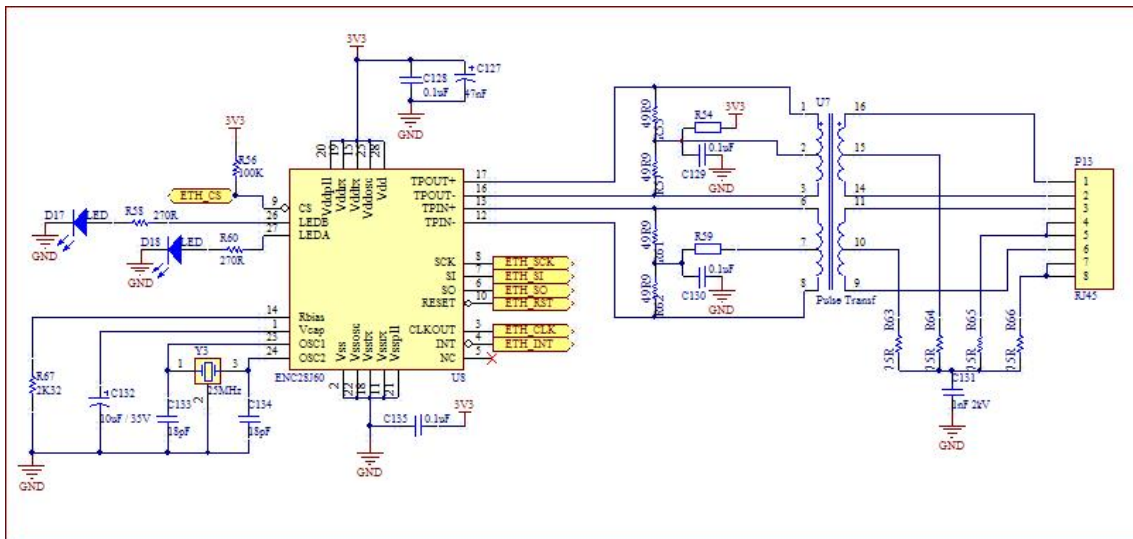



Figura 26 – Circuito do controlador Ethernet ligado ao *FPGA*.

Conforme já adiantado, este controlador utiliza interface *SPI*. Essa interface faz uso de quatro sinais para a comunicação serial entre dois dispositivos, onde um deles é o *master* e o outro *slave*. O *master* deve enviar o sinal de *clock*, pois é um protocolo síncrono. Outro sinal é para habilitação do *slave*, chamado de *Chip Select* (CSL), utilizado quando se tem vários dispositivos compartilhando o mesmo barramento de dados *SPI*. O terceiro é por onde trafegará os dados do *master* para o *slave* e o quarto é para o tráfego de dados na direção oposta.

Segue, na Tabela 19, a descrição dos pinos utilizados, bem como a função de cada um deles.

Tabela 19 – Descrição dos pinos e sinais utilizados no protocolo Ethernet - *FPGA*.

NOME	***TIPO***	***DESCRIÇÃO***	***PINO NO ARM***
SCK	O	*Clock* para a interface *SPI*	178
SI	I	*Data In* para interface *SPI*	162
SO	O	*Data Out* para interface *SPI*	163
RST	I	Sinal de *reset*	164
CS	I	*Chip select* é utilizado para selecionar o *chip* quando se tem mais de um dispositivo interconectado à interface *SPI*. Se colocado em nível lógico 1 o dispositivo desabilita a comunicação com o microcontrolador;	165
CLKOUT	O	Pino de *clock* de saída programável	180
INT	O	*Interrupt* é utilizado para notificar o microcontrolador quando não se opta pelo *pooling*.	167

4.3.10 MÓDULO PRINCIPAL DO FPGA SPARTAN 3 (FPGA-CORE)

Assim como no módulo *ARM-CORE*, o módulo *FPGA-CORE* também agrega apenas os componentes puramente vinculados ao *FPGA*, essenciais para seu pleno funcionamento. Por ter muitos pinos a serem conectados, no total 208, a apresentação do componente é feita por blocos: (a) alimentação, (b) SPI, (c) Bank 0, (d) Bank 1, (e) Bank 2 e (f) Bank 3.

No que concerne à alimentação, foram utilizados inúmeros capacitores de filtro de sinais para estabilizar cada tensão de entrada. Conforme já explicado, cada um dos quatro bancos de entradas e saídas podem ser alimentados com alguns valores pré-definidos de tensão. De forma a preservar esta característica e dar, ao usuário, a opção de escolha foram inseridos seletores, *jumpers*, que possibilita alimentar cada banco com 1V2, 2V5 ou 3V3, independentemente. O seletor P4 viabiliza a seleção de alguma das 3 tensões

para o Banco 0. Da mesma forma, o seletor P5 está para o Banco 1, P7 para o Banco 2 e, por fim, o P6 para o Banco 3. A Figura 27 mostra, esquematicamente, o bloco de alimentação do *FPGA*.

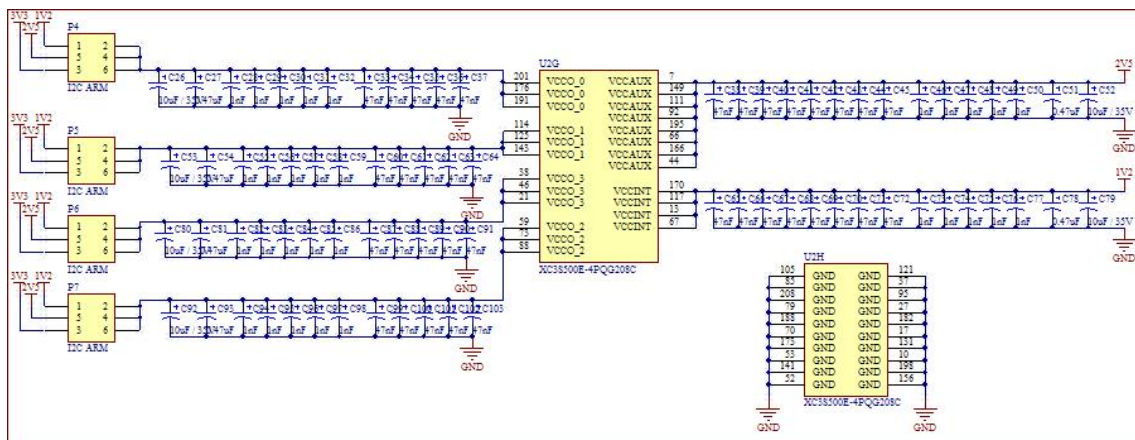



Figura 27 – Bloco de alimentação do *FPGA*.

Seguindo recomendações feitas por colegas atuantes no desenvolvimento de *PCBs* no mercado foram inseridos conjuntos de 5 capacitores de 1nF, 8 de 47nF, um de 0.47uF e um de 10uF, em paralelo com cada entrada de alimentação, de modo a reduzir a possibilidade de oscilação da fonte de tensão que será utilizada para alimentar o *FPGA*.

Em seguida, foram feitas as ligações de um outro bloco que é apresentado separadamente, o da comunicação *SPI*. Esta interface é compartilhada entre a memória externa ligada ao *FPGA* e também com o *ARM* para que ambos tenham mais um meio de troca de dados.

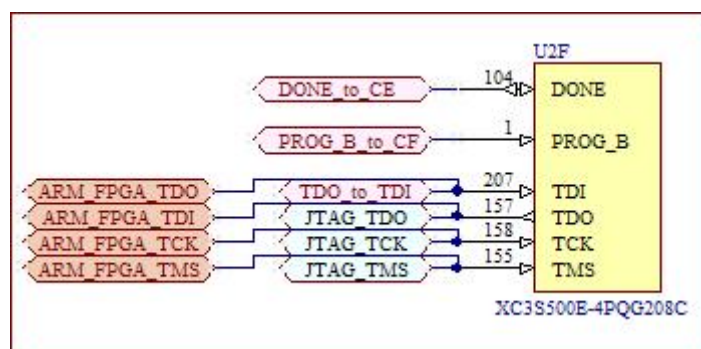



Figura 28 – Bloco SPI do *FPGA*.

Posteriormente, foram feitas as conexões referentes ao *Bank 0*, que corresponde à borda superior do *FPGA*. Neste bloco, foram inseridas as conexões com o controlador *Ethernet* bem como alguns pinos para comunicação com o *ARM* via *I*^{2}*C* e *SPI*. Os demais pinos, em todos os *Banks*, identificados como *PINxxx*, são disponibilizados em barras de pinos para que o

usuário possa utilizá-los como queira. A Figura 29 mostra a disposição dos pinos do *Bank 0*.

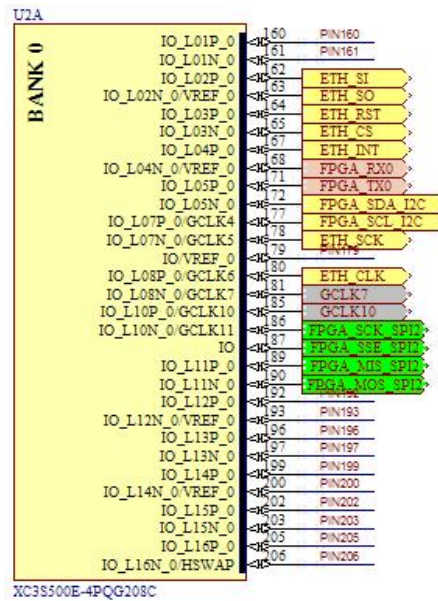



Figura 29 – Bloco Bank 0 do *FPGA*.

O *Bank 1*, situado na borda direita do dispositivo, pode ser inteiramente utilizado pelo usuário, vez que todos os pinos são disponibilizados em barras de pinos laterais à placa. A Figura 30 mostra a disposição dos pinos do *Bank 1*.

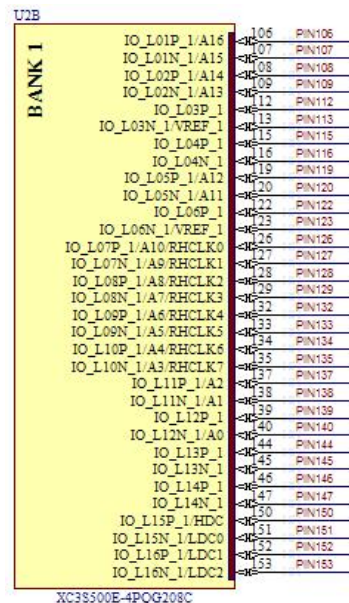



Figura 30 – Bloco Bank 1 do *FPGA*.

Grande parte do *Bank-2* foi utilizado para interface de botões e *LEDs*, conforme já descrito no subtópico 4.3.3. Além dos botões e *LEDs*, este banco ainda contêm ligações diretas nas entradas de interrupção do *ARM*, ligações diretas nos pinos de transmissão (T_x) e recepção (R_x) do *ARM*, bem como para a

comunicação *JTAG* e memória externa. Situado na borda inferior do dispositivo, o *Bank-2* ainda possui algumas entradas/saídas disponibilizadas em barras de pinos laterais à placa para livre uso pelo usuário.

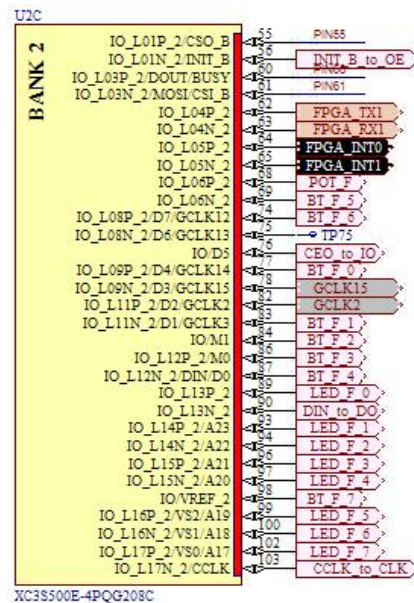



Figura 31 – Bloco Bank 2 do *FPGA*.

O *Bank-3*, situado na borda esquerda do dispositivo, pode ser inteiramente utilizado pelo usuário vez que todos os pinos deste barramento são disponibilizados em barras de pinos laterais à placa.

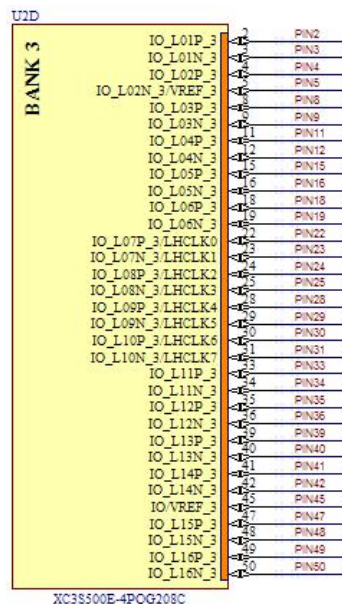



Figura 32 – Bloco Bank 3 do *FPGA*.

4.3.11 PROJETOS DE INTEGRACAO ENTRE OS SUBSISTEMAS ARM E FPGA

Após a montagem individual do processador *ARM*, do *FPGA* e respectivos módulos, restou então fazer a conexão entre os dois núcleos de processamento.

Pelo fato de os componentes utilizados serem SMD, os primeiros testes de integração só foram possíveis de serem realizados após a plataforma pronta. Com isso, a equipe do projeto não sabia de antemão como essa integração funcionaria, se precisaria de alguma interface de comunicação, se precisaria utilizar algum protocolo ou se funcionaria de maneira direta, conexão pino-a-pino. Pela dúvida, optou-se por realizar a conexão pino a pino utilizando as saídas destinadas comunicações via protocolo RS232, I^2C, SPI e JTAG. Dessa forma, caso a comunicação não funcionasse pino-a-pino ter-se-ia a opção de testá-la via protocolos.

Conforme já descrito, todos os *Banks* do *FPGA* podem ser alimentados com tensões de 1V2, 2V5 e 3,3V. Alimentando-os com 3V3 torna a comunicação compatível, em nível de tensão, pois o *ARM* também é alimentado com este nível de tensão. Algumas ligações que já estão estabelecidas utilizam o *Bank 0* e o *Bank 2* mas os demais pinos dos outros bancos também podem ser utilizado para a comunicação direta entre o processador *ARM* e o *FPGA*.

Outro cuidado a ser tomado durante o processo de comunicação entre os dois núcleos é a correta configuração dos pinos de cada um. Por exemplo, se a comunicação estiver sendo estabelecido no sentido do ARM para o FPGA, os pinos do ARM dedicados à essa troca de dados devem ser configurados com saída e os do FPGA como entrada, e vice-versa. Com isso, evita-se eventuais choques em nível de tensão causando curto circuito em ambos os componentes. Evita-se ainda incompatibilidades, em tensão, entre um componente ativo e outro em *tri-state*[25], por exemplo.

A seguir, na

Tabela 20, é mostrada a correlação dos pinos utilizados para a integração dos sistemas.

[25] Em eletrônica digital, portas lógicas com saídas tri-state ou **3-**state permitem a geração de valores de 0, 1 ou Z. Uma saída *Z* pode ser considerada como uma saída desconectada do resto do circuito, pois se apresenta em um estado de alta impedância. A intenção deste estado é permitir diversos circuitos a compartilharem da mesma linha ou barramento de dados, sem afetar umas as outras.

Tabela 20 – Ligações de integração entre *ARM* e *FPGA*.

NOME	***PROTOCOLO***	***PINO NO ARM***	***PINO NO FPGA***	***BANK***
*FPGA*_Rx0	*Serial*	21	168	0
*FPGA*_Tx0	*Serial*	19	171	0
*FPGA*_Rx1	*Serial*	34	63	2
*FPGA*_Tx1	*Serial*	33	62	2
*FPGA*_INT0	*Interrupção*	46	64	2
*FPGA*_INT1	*Interrupção*	45	65	2
*FPGA*_SCL_I^2C	0^2C	37	177	0
*FPGA*_SDA_I^2C	I^2C	41	172	0
*FPGA*_SCK_SPI2	*SPI*	47	186	0
*FPGA*_MIS_SPI2	*SPI*	53	187	0
*FPGA*_MOS_SPI2	*SPI*	54	189	0
*FPGA*_SSE_SPI2	*SPI*	55	190	0
*ARM_FPGA*_TDO	*JTAG*	64	157	-
*ARM_FPGA*_TDI	*JTAG*	60	207	-
*ARM_FPGA*_TCK	*JTAG*	56	158	-
*ARM_FPGA*_TMS	*JTAG*	52	155	-

4.3.12 DIAGRAMA DE BLOCOS DO PROJETO DE INTEGRAÇÃO

Ao final da montagem de todos os circuitos foi elaborado um diagrama de blocos para melhor visualização do sistema. Cada retângulo representa um bloco ou subsistema. Os retângulos centrais representam cada unidade de processamento, *ARM* e *FPGA*. Na lateral externa de cada unidade de processamento encontram-se os ou módulos que dão suporte ao respectivo núcleo.

O diagrama também mostra os sinais de entradas e saídas de cada módulo. Cada linha ligando os módulos representa um sinal de entrada, saída ou entrada e saída dos módulos. O nome dado a cada linha busca remeter à função do sinal ali trafegado. A Figura 33 mostra esquematicamente o diagrama de blocos do sistema.

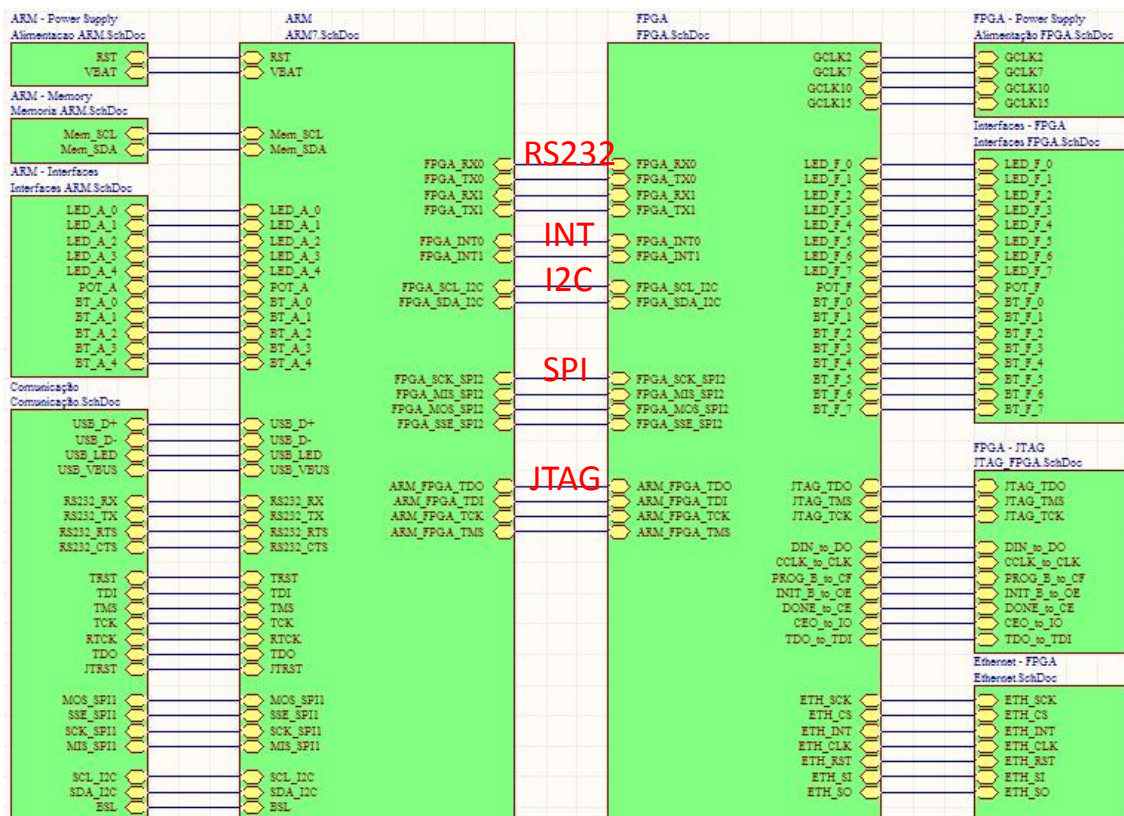



Figura 33 – Esquemático dos módulos constituintes do sistema de controle.

4.4 SELEÇÃO E COMPRA DE COMPOENTES

Em conjunto com a elaboração dos circuitos foi feita a seleção dos componentes por meio de análise de disponibilidade de fornecimento, facilidade de aquisição, confiabilidade, quantidade em estoque, continuidade de fabricação, valor em relação aos similares e compromisso ambiental de modo a realizar a melhor escolha. junto a alguns fornecedores nacionais e internacionais. Entre os fornecedores nacionais, foram contatados:

a. ME Componentes - Webpage: www.mecomp.com.br (Brasília)
b. Sitecomp - webpage: www.sitecomp.com.br (São Paulo)
c. RGN Componentes - Email: vendas.rgncomponentes@gmail.com.br (São Paulo)

Entre os fornecedores internacionais, foram contatados:

a. Farnell - webpage: www.farnell.com.br (EUA)
b. Mouser - Webpage: www.mouser.com (EUA)
c. Digikey - webpage: www.digikey.com (EUA)

Pelo conjunto de componentes selecionados considerando a facilidade de aquisição, capacidade de fornecimento e tendo como referência o menor custo optou-se por realizar toda a compra de um único fornecedor norte-americano, a Digikey Corp.. Foram comprados componentes em quantidade equivalente à montagem de três plataformas. O APÊNDICE D apresenta a lista completa dos componentes utilizados para a montagem de cada unidade da plataforma.

4.5 PROJETO DA PLACA DE CIRCUITO IMPRESSO

Seguindo as definições metodológicas, optou-se pela contratação de um profissional com experiência na elaboração de layout de PCI. Seguindo recomendações de colegas engenheiros atuantes na elaboração de projetos eletrônicos, foi contratado o *layoutista* Paulo Egon para a execução do serviço. Ele executa projetos de grandes empresas nacionais26 e é bem reconhecido pela qualidade do seu trabalho. Contato: PCB Projetos Eletrônicos: pcbproj@gmail.com.

4.6 MANUFATURA DA PLACA DE CIRCUITO IMPRESSO

Seguindo as definições metodológicas, optou-se pela contratação de uma empresa especializada nesse tipo de serviço. Por meio de uma busca na internet foram encontradas diversas empresas nacionais que realizam a manufatura de PCI entretanto como o projeto ora em tela contem quatro camadas, dentre as contatadas, apenas quatro tinham ferramental para executar o serviço. São elas:

a. Circuibras – website: http://www.circuibras.com.br (Curitiba - PR)
b. Micropress – website: http://www.micropress.com.br (São Paulo - SP)
c. PCI Paraná – website: http://www.pciparana.com.br (Pinhais - PR)
d. Print CI – website: http://www.printcir.com.br (Taboão da Serra - SP)

Após uma analise de custos foi selecionada a empresa *Print Circuits Eletronica Ltda* para a impressão de quatro placas conforme projeto desenvolvido. A placa foi elaborada utilizando quatro camadas e dimensões retangulares de 160x150mm, conforme exposto no APÊNDICE B.

---

[26] Intensionalmente mantidas em sigilo.

No caso em tela, foi escolhida, para compor o substrato da PCI, uma combinação entre os materiais FR-1/FR-2 e FR-4, pois combina bom desempenho em sistemas de placas rígidas, alta durabilidade e baixo custo.

4.7 MONTAGEM DOS COMPONENTES

Conforme definido na metodologia de desenvolvimento, esta etapa foi terceirizada uma vez que exige infraestrutura, ferramental e experiência das quais a equipe não dispõe. O objetivo da contratação foi viabilizar a obtenção da plataforma e ainda evitar possibilidades de danos aos componentes e consequentemente ao sistema como um todo.

Após uma busca na internet por empresas especializadas no serviço foi feita a cotação com três delas:

a. TRACE LAY-OUT – website: http://www.tracelayout.com.br (São Paulo - SP)
b. ComLink Equip. Eletronicos – website: http://www.comlink.ind.br/ (Caxias do Sul - RS)
c. Equitronic – website: http://www.equitronic.com.br (Campinas - SP)

Após uma análise de custos e confiabilidade, foi contratada a empresa *Equitronic Equipamentos Eletrônicos Ltda*, em São Paulo. A empresa está no mercado há mais de 16 anos, já executou serviços para os mais diversos tipos de clientes nacionais[27].

4.8 TESTES DOS SISTEMAS

A fase de testes do sistema , bem como os resultados obtidos, estão descritos detalhadamente no CAPÍTULO 5.

4.9 CONCLUSÃO DO CAPÍTULO

Neste capítulo, foram apresentadas todas as etapas percorridas durante o desenvolvimento da plataforma. Aqui, encontram-se documentados todos os circuitos eletrônicos que foram efetivamente implementados na versão 1.1 da plataforma, conforme metodologia definida e apresentada no CAPÍTULO 3.

[27] Disponível em: http://www.equitronic.com.br/. Acesso em: 01/01/2012.

Adicionalmente, foram apresentados todos os módulos de suporte aos núcleos de processamento bem como a justificativa de escolha de cada circuito. Ao final desta etapa, diversas revisões foram feitas, pela equipe interna e colaboradores externos[28], de modo a aprimorar os circuitos desenvolvidos e eliminar possíveis falhas. Algumas das contribuições e mudanças de projeto advêm da experiência prática que a equipe possui.

Buscando seguir a metodologia elaborada, os projetos foram desenvolvidos de maneira independente uns dos outros para que ao final pudessem ser reunidos em um sistema integrável, harmônico, de fácil entendimento e manutenção.

Foi relatado ainda que a execução das etapas finais de projeto, fabricação de placas e montagem de componentes tiveram de ser terceirizadas por empresas nacionais por exigirem infraestrutura, ferramentas e maquinário especializado.

Ao final desta etapa, chegou-se à materialização da plataforma propriamente dita. A etapa de desenvolvimento consistiu na transformação dos projetos em produto pronto a ser testado e experimentado.

O APÊNDICE B é destinado à apresentação da plataforma física e seus subsistemas.

[28] Engenheiro Rodrigo Willians De Carvalho – Graduado no Curso de Engenharia Mecatrônica da UnB (2006) e MsC Magno Batista Corrêa, mestre em Sistemas Mecatrônicos pela UnB (2011).

CAPÍTULO 5
TESTES REALIZADOS E RESULTADOS OBTIDOS

A metodologia adotada para testes da plataforma seguiu a lógica de desenvolvimento dela. Por terem sido utilizados sistemas modulares, desacopláveis por meio de *jumpers*, os testes seguiram na linha de desenvolvimento deles, ou seja, inicialmente verifica o funcionamento do módulo isolado, quando possível, e em seguida integra-o ao restante do sistema.

Dessa forma, é possível realizar testes com segurança, pois evita-se que eventuais falhas em um módulo possam prejudicar todo o sistema. Caso sejam identificadas falhas elas são sanadas antes do módulo ser integrado ao restante da plataforma. A seguir, será apresentada a plataforma desenvolvida e seus principais módulos (descrição mais detalhadas sobre a disposição dos componentes, disponível no APÊNDICE B). Em seguida, são descritas todas as etapas seguidas durante a fase de testes bem como os resultados obtidos.

Mais detalhes sobre metodologia adotada para os testes realizados, vide CAPÍTULO 3. Foram utilizadas diversas ferramentas de auxilio à programação e testes, tanto para o *ARM* quanto para o FPGA. Todos os procedimentos de configuração de cada uma dessas ferramentas estão descritos no *APÊNDICE E.* Os códigos de programação utilizados estão disponíveis no APÊNDICE F.

5.1 APRESENTAÇÃO DA PLATAFORMA

De forma a ilustrar os resultados alcançados ao longo do trabalho a seguir são apresentados, de forma resumida, a plataforma em como o posicionamento dos principais módulos de cada núcleo.

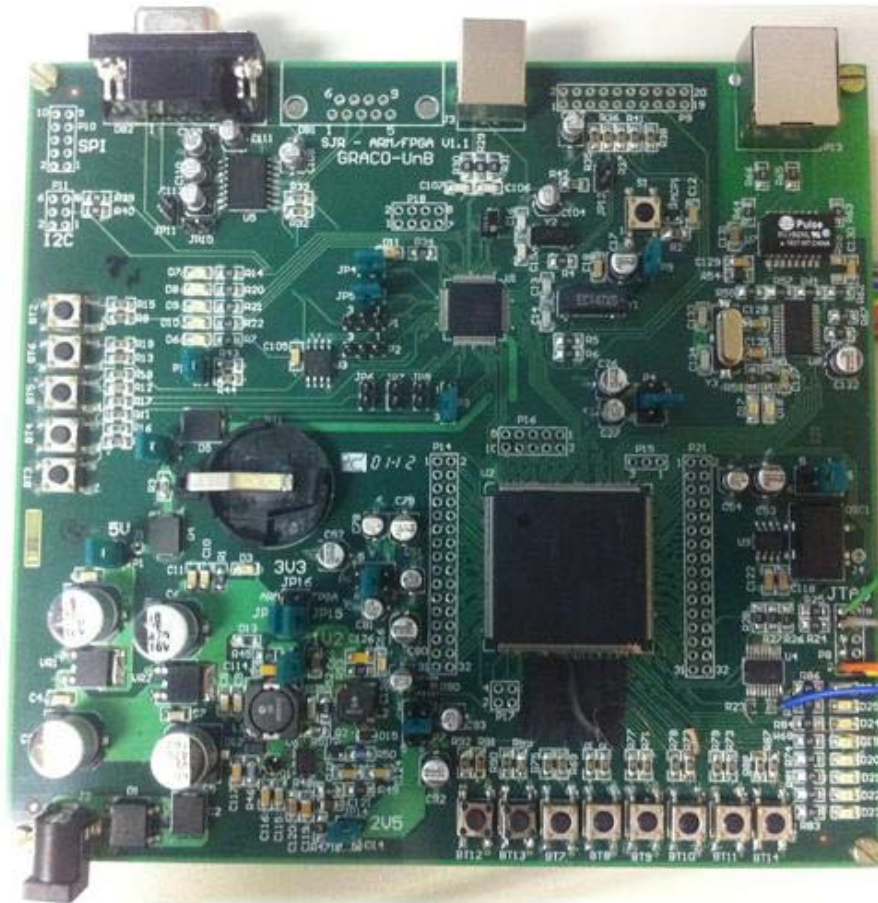



Figura 34 – Foto da plataforma reconfigurável montada, pronta para testes.

5.1.1 NÚCLEO DO MÓDULO *ARM*

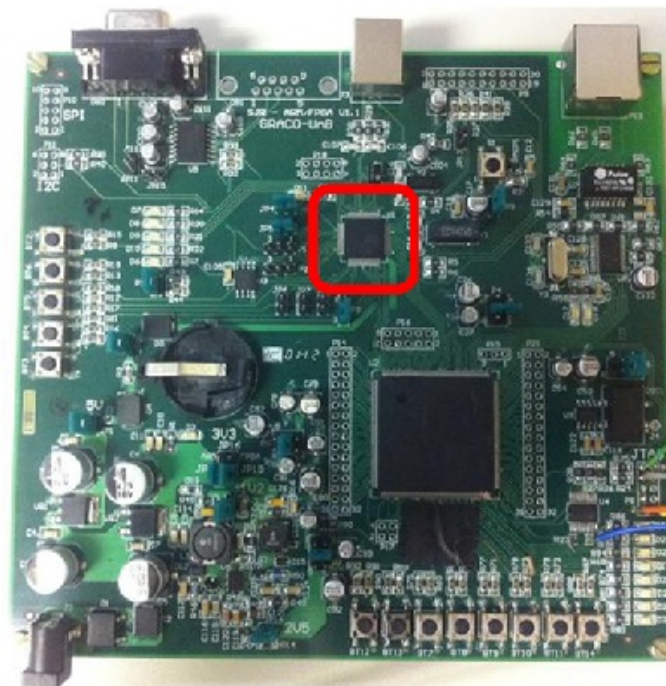



Figura 35 – Núcleo do módulo ARM, referenciado na plataforma reconfigurável.

5.1.2 MÓDULO DE ALIMENTAÇÃO DO *ARM*

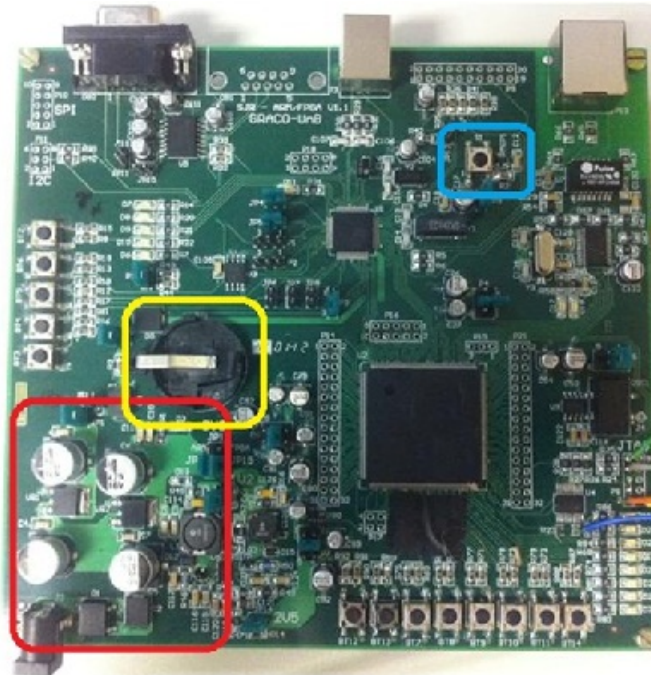



Figura 36 – Regulador de tensão (vermelho), bateria (amarelo) e *driver reset* (azul), referenciados na plataforma reconfigurável.

5.1.3 MÓDULO DE MEMÓRIA DO *ARM*

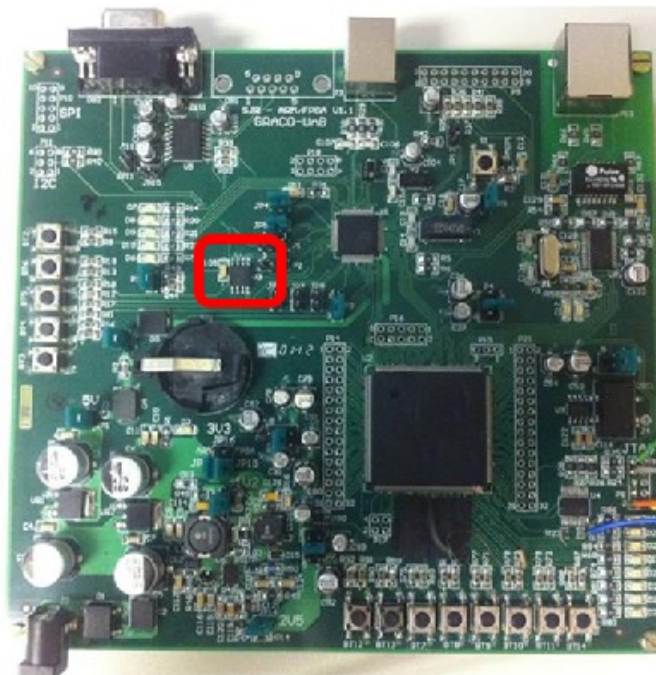



Figura 37 – Memória ligada ao *ARM,* referenciada na plataforma reconfigurável.

5.1.4 MÓDULO DE INTERFACES DO *ARM*

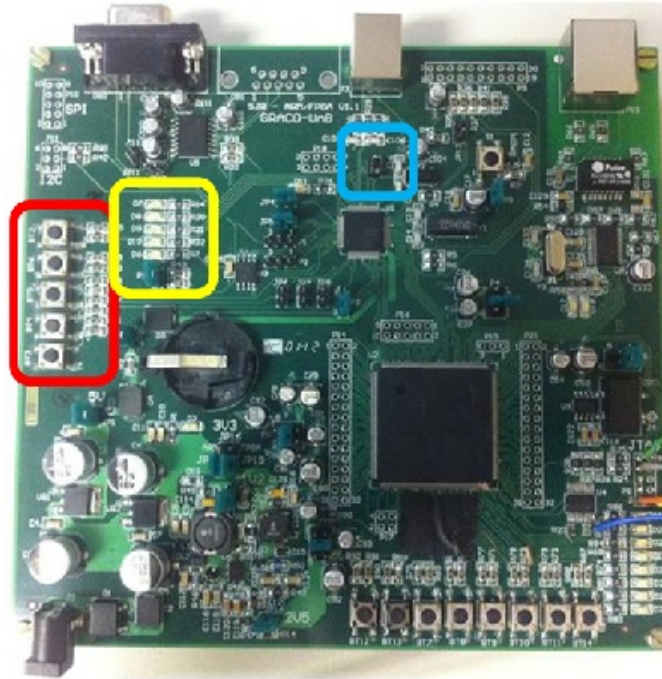



Figura 38 – Botões (vermelho), *LEDs* (amarelo) e potenciômetro (azul) ligados ao *ARM,* referenciados na plataforma reconfigurável.

5.1.5 MÓDULO DE COMUNICAÇÃO EXTERNA DO *ARM*

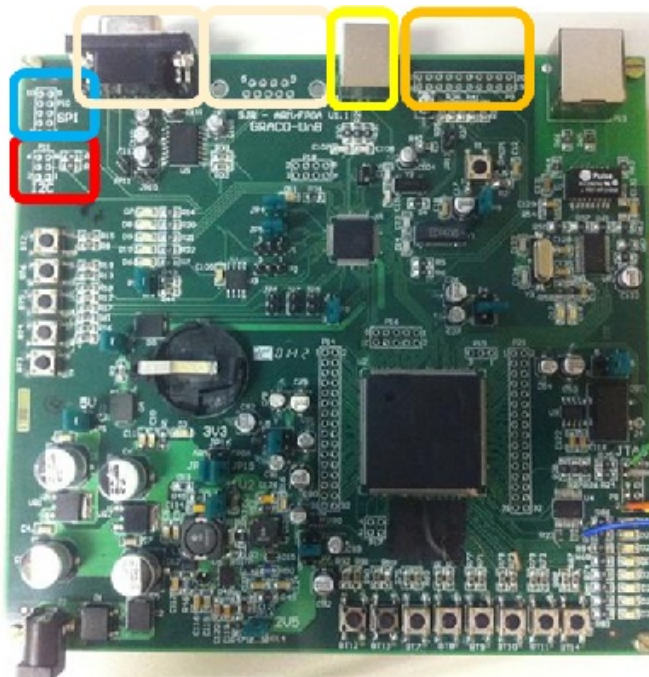



Figura 39 – Da esquerda (vermelho) para a direita (laranja) tem-se os conectores: *I^2C*, *SPI, RS232_1, RS232_2, USB e JTAG* ligados ao *ARM,* referenciados na plataforma reconfigurável.

5.1.6 NÚCLEO DO MÓDULO *FPGA*

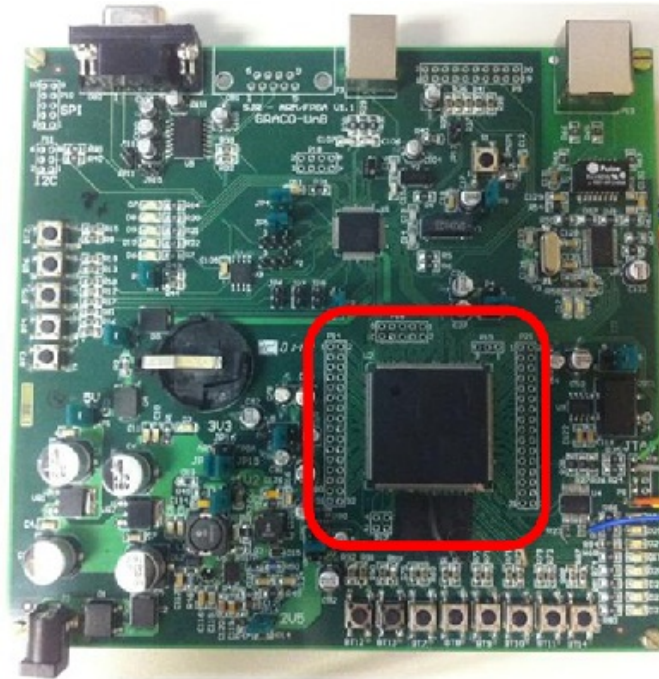



Figura 40 – Núcleo do módulo *FPGA,* referenciado na plataforma reconfigurável.

5.1.7 MÓDULO DE ALIMENTAÇÃO DO *FPGA*

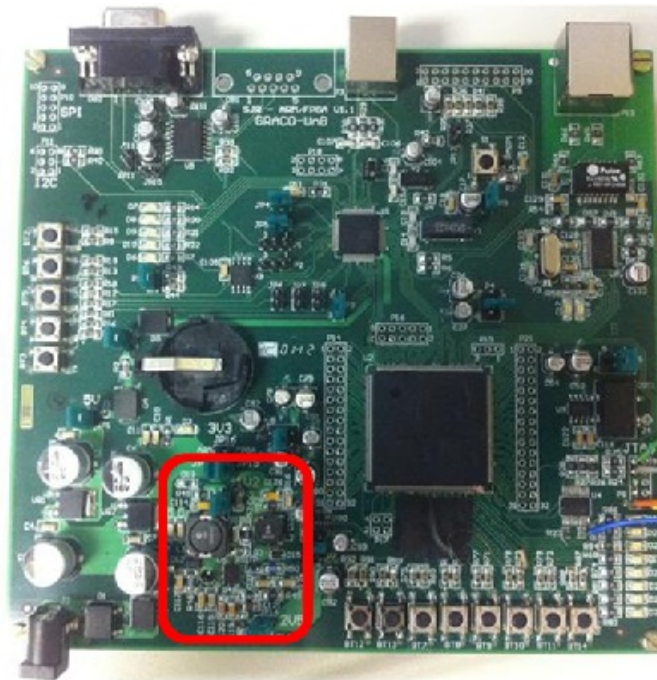



Figura 41 – Módulo de alimentação do *FPGA,* referenciado na plataforma reconfigurável.

5.1.8 MÓDULO DE MEMÓRIA EXTERNA DO *FPGA*

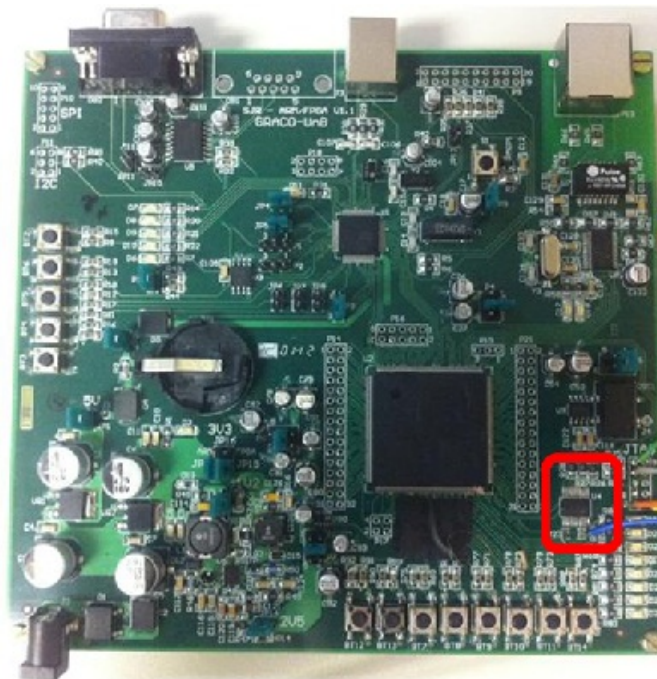



Figura 42 – Módulo de memória externa do *FPGA,* referenciada na plataforma reconfigurável.

5.1.9 MÓDULO DE INTERFACES DO *FPGA*

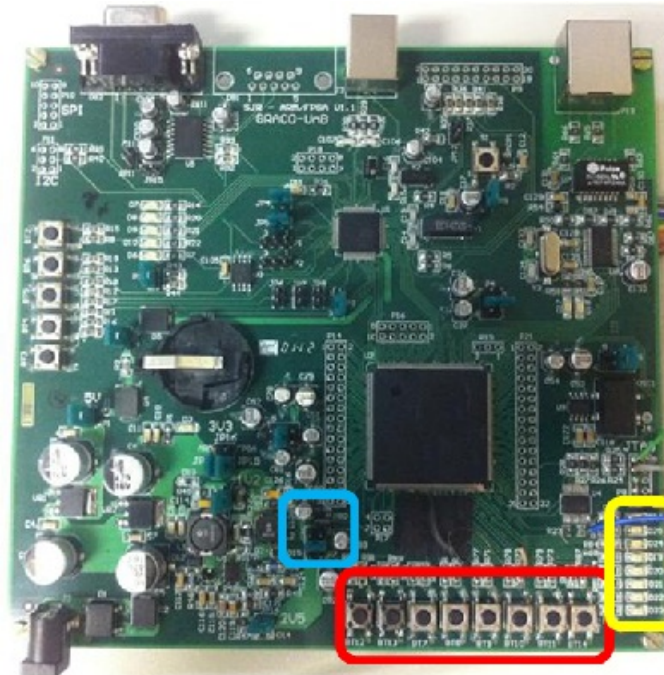



Figura 43 – Botões (vermelho), *LEDs* (amarelo) e potenciômetro (azul) ligados ao *FPGA,* referenciados na plataforma reconfigurável.

5.1.10 MÓDULO DE COMUNICAÇÃO EXTERNA DO *FPGA*

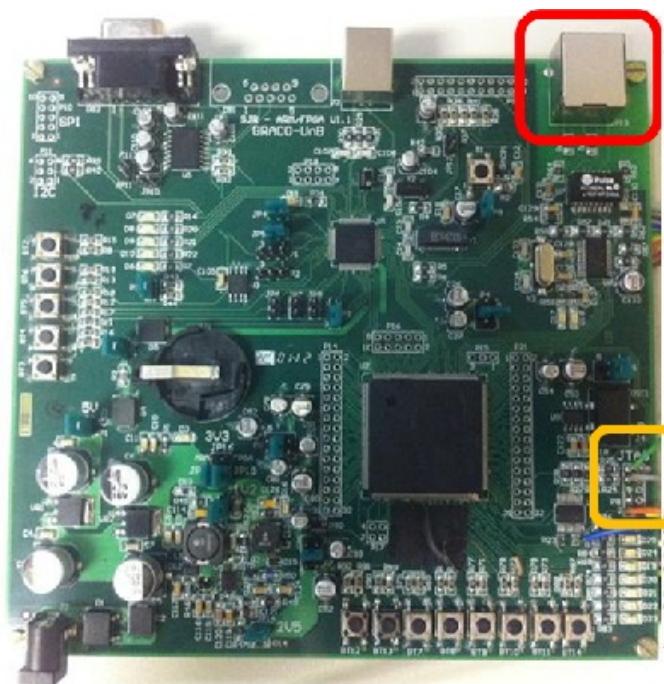



Figura 44 – Conectores *ethernet* (vermelho) e *JTAG* (amarelo) ligados ao *FPGA,* referenciados na plataforma reconfigurável.

5.2 TESTE DE *HARDWARE – PCI*

O primeiro teste a que a plataforma foi submetida consistiu na verificação da boa condutibilidade das trilhas, áreas de isolamento (*clearance*), assim como a ausência de curtos-circuitos na placa de circuito impresso. Este teste foi realizado assim que a placa foi impressa e finalizada, pela própria empresa fabricante da *PCI*.

O teste precisa ser realizado na *placa crua*, ou seja, antes mesmo de serem fixados os componentes. Dessa forma, caso haja alguma anomalia ou conexão com defeito poderão ser identificados antes da montagem da plataforma.

Essa identificação precoce objetiva poupar a equipe de prosseguir o desenvolvimento do projeto utilizando uma base defeituosa. Problemas desse tipo podem tomar muito tempo da equipe por meio da revisão minuciosa das conexões e circuitos que mesmo bem elaborados não funcionarão adequadamente. Além disso, onera o projeto pela montagem dos componentes e pelo tempo extra, imposto para a finalização do projeto.

Fabricantes responsáveis se obrigam a fazer esse tipo de teste, e só liberam o produto ao cliente caso esteja de acordo com o projeto contratado. No caso hora apresentado, foi escolhido um fabricante com experiência na elaboração de placas *PCI* que se responsabiliza pela perfeita execução do projeto. Mesmo assim, ao se receber as placas, varias conexões foram conferidas, especialmente aquelas relacionadas à alimentação. No presente caso, nenhuma anomalia foi identificada.

5.3 TESTES DOS MÓDULOS SEPARADOS DO SISTEMA

Partiu-se então para realizar os testes de funcionamento da plataforma, ou seja, cada módulo foi verificado a fim de encontrar uma eventual inconsistência ou discrepância com o planejado.

5.2.1 TESTE DA FONTE DE ALIMENTAÇÃO DO ARM

Uma fez com a plataforma montada, partiu-se para os primeiros testes de funcionamento. Ainda sem a utilização de qualquer *jumper* ligou-se a plataforma na rede elétrica a fim de verificar a tensão gerada na saída dos reguladores de tensão, LM7805 e LM7833 (Vide sessão 4.3.1.1). Como ainda estavam totalmente desconectados ao restante da plataforma, mesmo que apresentassem defeitos não prejudicariam o restante dos componentes.

As tensões lidas no osciloscópio foram exatamente como planejadas: 9, 5 e 3.3V. Depois de constatado o seu correto funcionamento, ligou-se o *jumper* JP2 para que a tensão 3V3A chegasse ao *ARM7*. Verificou-se novamente as tensões nos pinos de alimentação do *ARM7* e constatou-se que estavam em níveis normais de operação.

Como precaução, a alimentação da plataforma foi mantida por cerca de 10 minutos sem qualquer outro procedimento para verificar se algum dos reguladores apresentaria sinais de superaquecimento. Por meio de verificações sensitivas constatou-se que não sofriam aquecimento relevante. Partiu-se então para o processo de programação do *ARM7*.

5.2.2 TESTE DE PROGRAMAÇÃO DO ARM

Os testes de programação do *ARM* objetivaram a constatação de que o dispositivo estava operacional. Então, caso a equipe conseguisse carregar qualquer rotina no *ARM* e verificar que ele responde adequadamente aos comandos já seria suficiente para cumprir o objetivo estabelecido.

Optou-se neste caso, seguindo a metodologia de teste proposta no CAPÍTULO 3, por desenvolver uma rotina a qual utiliza não apenas as funcionalidades do *ARM7*, mas também testa parte do *hardware* de entrada e saída da plataforma. A rotina consistiu em programar o *ARM* para ler permanentemente as entradas associadas aos botões, e caso algum fosse pressionado, como resposta, o *ARM* enviaria um sinal para uma saída correspondente a um dos *LEDs* fazendo-o ligar.

Com este teste também é possível verificar o funcionamento da interface de programação *JTAG*, o circuito da saída RS232 e ainda o de *Reset*, implementados na plataforma.

Ao iniciar a fase de programação, uma dificuldade deparada foi encontrar o correto endereçamento dos pinos do *ARM7* de modo a configurá-los como, por exemplo, entrada e saída de *LEDs* e botões. A solução encontrada foi alterar o endereço, em hexadecimal, dividindo o conjunto em grupos de 2 até se encontrar o endereço desejado. A verificação foi feita por meio de leituras nos terminais de saída, tanto dos botões quanto dos *LEDs*. O endereçamento encontrado pode ser visualizado no código de testes utilizado, disponível no APÊNDICE F.

Neste caso, foi constatado que todos os circuitos acima descritos estavam em funcionamento pleno. Toda a plataforma respondeu conforme se esperava, desde a programação até o acendimento dos *LEDs* correspondentes aos botões.

Como forma de confirmar os resultados visuais, varias leituras foram feitas em diversos pinos do *ARM* para constatar se os níveis de tensão estavam variando conforme se esperava. Observou-se que os sinais estavam muito bem definidos com nível baixo em 0,3mV e nível alto em 3,3V.

Foi observado também que os reguladores de tensão não sofriam variação de temperatura relevante mesmo após longos períodos de teste (3 a 4 horas de programação) contínuos da plataforma. Dessa forma verifica-se a robustez da fonte de alimentação projetada para o *ARM*.

Portanto, por meio deste teste, constatou-se que os circuitos relacionados ao *ARM7* estão tecnicamente operacionais e prontos para serem utilizados nos mais diversos projetos.

A Figura 45 ilustra o resultado do teste. Ao ser pressionado o botão, o *LED* correspondente é aceso.

Figura 45 – Foto do teste de funcionamento do *ARM*.

5.2.3 TESTE DA FONTE DE ALIMENTAÇÃO DO FPGA

Verificado o funcionamento dos circuitos relacionados ao *ARM* partiu-se para a checagem do lado do *FPGA*. Assim como feito no *ARM*, a primeira etapa que se seguiu foi a verificação do funcionamento da fonte de alimentação do *FPGA*.

Conforme já descrito na sessão 4.3.1.1 o circuito de fornecimento das três tensões necessárias ao adequado funcionamento do *FPGA* gira em torno de apenas um componente, o *TPS75003*. Assim, bastou verificar a tensão de saída em cada terminal para saber se o circuito implementado estava ou não correto.

Inicialmente, foi observado que as tensões de 2V5 e 3V3 estavam bem estabilizadas, entretanto a de 1V2 não estava funcionando adequadamente. Como a fonte ainda estava desconectada do restante da plataforma essa falha não interferiu nos demais componentes. Partiu-se então para a revisão do circuito e constatou-se que o *mosfet Q1* (*SI2323DS*), associado à fonte 1V2, estava com o dreno e a fonte invertidos. Isso foi resultado de desatenção no momento do projeto esquemático ao copiar outro *mosfet* (Q2) e colá-lo sem revisar a pinagem.

De forma a não inutilizar a placa, optou-se por adaptar uma nova configuração para solucionar o problema. A solução se mostrou bem simples, bastou ressoldar o mesmo *mosfet, agora,* de cabeça para baixo fazendo com que seus pinos fossem reposicionados para a nova configuração.

Feita essa mudança, repetiu-se os testes de alimentação. Constatou-se que as tensões lidas no osciloscópio foram exatamente como planejadas: 1,2V; 2,5V e 3,3V. Foi observado que nenhum componente foi danificado uma vez que não houve carga que pudesse sobrecarregar a fonte. Até mesmo o próprio *mosfet* pôde ser reutilizado e apresentou respostas normais de operação.

Depois de verificado o seu correto funcionamento, foram inseridos *jumpers* em JP13, JP14 e JP15 para que as respectivas tensões chegassem ao *FPGA*. Verificou-se novamente as tensões nos pinos de alimentação do *FPGA* e constatou-se que estavam em níveis normais de operação.

Como precaução, a alimentação da plataforma foi mantida por cerca de 10 minutos sem qualquer outro procedimento para verificar se o regulador de tripla tensão apresentava algum sinal de superaquecimento. Por meio de verificações sensitivas constatou-se que não sofria aquecimento considerável. Partiu-se então para o processo de programação do *FPGA*.

5.2.4 TESTE DE PROGRAMAÇÃO DO FPGA

Garantida uma alimentação adequada no *FPGA*, passou-se à fase de programação do dispositivo. Como já introduzido ao final da sessão 4.3.7, ao se fazer os primeiros testes de gravação, ainda no *iMPACT*, percebeu-se que o pacote *ISE* não reconhecia o *FPGA* e tampouco a memória associada a ele. Pela revisão do circuito utilizado, observou-se que a plataforma tinha sido projetada para funcionar na configuração *Master-Serial* dependente de uma memória EE*PROM*, seguindo o modelo de referência da Figura 46. Esse modelo é proposto na documentação de referência para Spartan-3E como nota de aplicação (Xilinx, 2008).

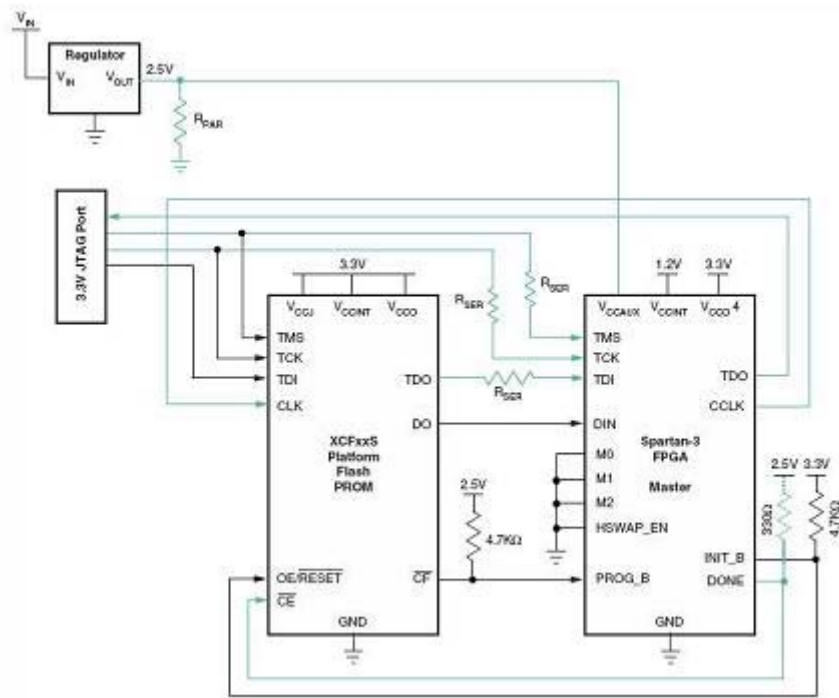



Figura 46 – Configuração utilizada na versão 1.1 para comunicação *FPGA*/FLASH[29].

Foi verificado que todas as ligações físicas da plataforma condiziam com as do projeto. Ao não se obter sucesso com o circuito na configuração acima, após várias tentativas, optou-se então por adaptar a plataforma a uma nova configuração, proposta na documentação de referencia do *XC3S500E* (vide Figura 47).

[29] Disponível em: http://www.xilinx.com/support/documentation/application_notes/xapp453.pdf. Acesso em: 27/06/2010.

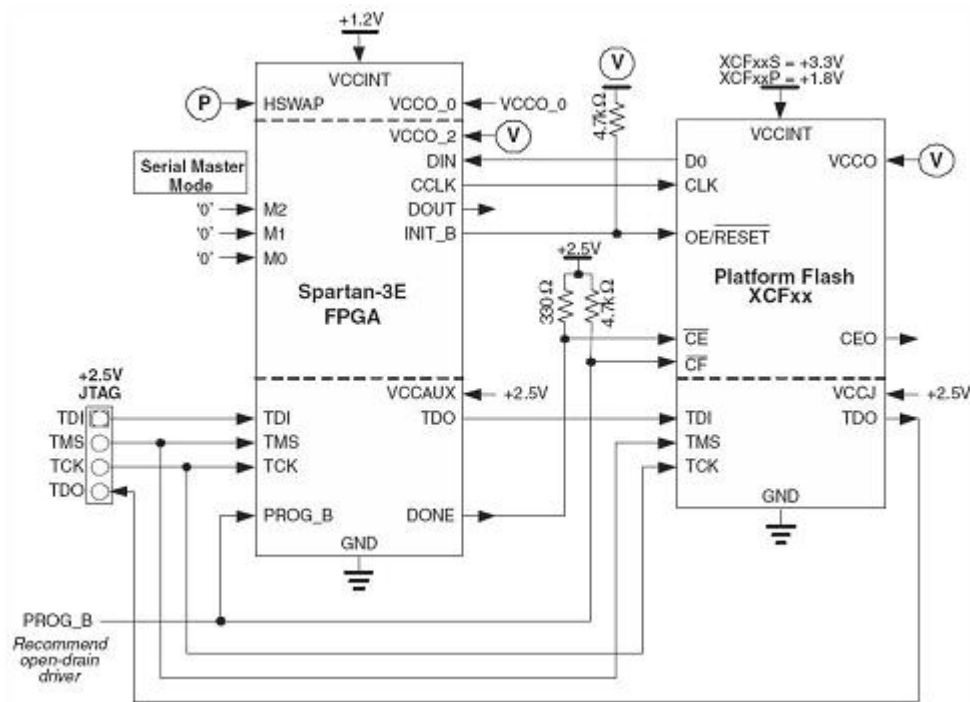



Figura 47 – Nova configuração proposta para comunicação *FPGA*/FLASH[30].

A principal mudança entre as duas configurações é o reposicionamento dos elementos *FPGA* e memória. No primeiro, os sinais de programação, vindos do computador via *JTAG*, passam primeiro pela memória para depois chegarem ao *FPGA*, já na segunda configuração eles vão direto ao *FPGA* para então seguirem para a memória.

Para se conseguir esse novo esquemático foi preciso alterar fisicamente algumas ligações e trilhas na plataforma, uma vez que esse problema foi detectado somente após a confecção da placa. Todavia, não foi preciso danificar a placa ou mesmo romper trilhas, bastou apenas adaptar umas soldas e buscar os sinais diretamente nos pinos das saídas desejavas por meios de fios condutores. Após a montagem da nova configuração o pacote de programação *ISE* reconheceu adequadamente o *FPGA* e a memória *EEPROM*.

Verificado o adequado funcionamento do sistema, seguiu-se com os testes de programação. Os testes de programação do *FPGA* objetivaram a constatação de que este lado da plataforma também se apresenta operacional.

Novamente, optou-se por desenvolver uma rotina na qual utiliza não apenas as funcionalidades do *FPGA,* mas também verificar parte do *hardware* de entrada e saída da plataforma. A rotina consistiu em configurar o *FPGA* para acender

[30] Disponível em: http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf. Acesso em: 27/06/2012.

os *LEDs* caso fosse pressionado cada botão correspondente. Vide código VHDL no APÊNDICE F.

Foi constatado que os circuitos utilizados estavam em funcionamento pleno. Toda a plataforma respondeu conforme as expectativas, desde a programação até o acendimento dos *LEDs* correspondentes aos botões.

Como forma de confirmar os resultados visuais, várias leituras foram feitas em diversos pinos do *FPGA* para constatar se os níveis de tensão estavam variando conforme se esperava. Observou-se que os sinais estavam muito bem definidos com nível baixo em 0,3mV e nível alto em 3,3V.

Por outro lado, foi observado também que o regulador de tensões não sofreu variação de temperatura relevante mesmo após longos períodos de teste contínuos da plataforma. Portanto, por meio deste teste, constatou-se que os circuitos relacionados ao *FPGA* estão tecnicamente operacionais e prontos para serem utilizados nos mais diversos projetos.

A Figura 48 ilustra o resultado do teste, ou seja, ao ser pressionado o botão, o *LED* correspondente é aceso.

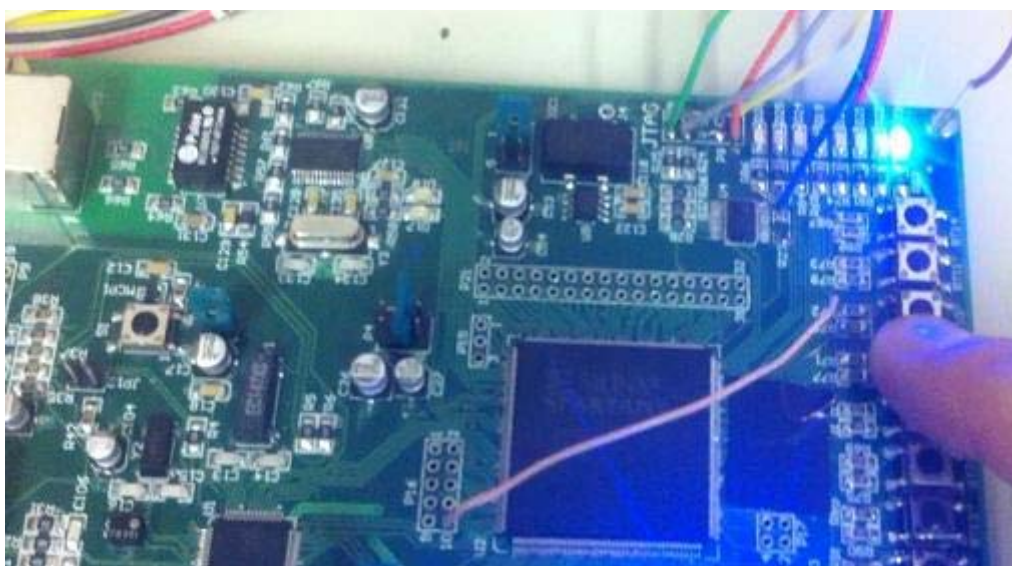



Figura 48 – Foto do teste de funcionamento do *FPGA*.

5.4 TESTE DE INTEGRAÇÃO (COMUNICAÇÃO ENTRE O *ARM* E O *FPGA*)

A última etapa de testes da plataforma consistiu em comprovar a possibilidade de integração entre os dois núcleos. A maior questão era saber como a comunicação deveria ser feita, se necessitaria de um protocolo, por exemplo,

para que o processador ARM e o FPGA pudessem receber e interpretar os sinais trocados entre si.

Como ambos são disponibilizados apenas em encapsulamento SMD, torna-se impraticável uma fase de testes em *protoboard*, por exemplo. Os primeiros testes precisaram ser feitos já com a plataforma pronta. Começando pelo mais simples, o primeiro teste foi utilizar uma das trilhas de ligação direta entre eles para estabelecer uma comunicação de uma via apenas. Para isso, foram aproveitadas as rotinas já desenvolvidas e testada em cada núcleo.

No *ARM*, fez-se apenas a modificação do pino de saída do sinal ao ser pressionado um botão que, em vez de acender um *LED*, agora é enviado para o *FPGA*. No *FPGA*, a modificação consistiu apenas na mudança do pino de entrada do sinal que, antes vindo do botão, agora provém do *ARM*. Uma vez carregadas as rotinas nos dois núcleos verificou-se que a comunicação foi estabelecida de forma direta e eficiente. Destaca-se ainda que o *Bank 0* do *FPGA*, utilizado na comunicação, foi alimentado em 3.3V compatibilizando assim com o nível de tensão utilizado em ambos os núcleos.

Em seguida, partiu-se para a configuração em que o *FPGA* envia o sinal para o *ARM*. Semelhante ao que foi feito para a via única, repetiu-se o procedimento para que a comunicação se estabelecesse nas duas vias, ou seja, pressionando um botão ligado ao *FPGA* consegue-se acender um *LED* ligado ao *ARM*. Visualmente, e por meio de leitura de tensões, foi possível identificar o funcionamento da plataforma conforme planejado.

Verificou-se que existe plena compatibilidade de comunicação entre os dois núcleos por via direta, sem necessidade de configuração especial ou protocolo específico (ressaltando que os *banks* utilizados no *FPGA* devem ser alimentados na mesma tensão de referência que o *ARM*). Os testes realizados comprovaram que é possível haver uma troca de informações nas duas vias, *FPGA* para *ARM* e *ARM* para *FPGA*. As rotinas geradas utilizaram apenas duas vias, uma para cada sentido de transmissão, apenas como prova de efetividade do circuito. Todas as doze vias podem ser usadas para a troca de dados entre ambos.

A Figura 49 ilustra o resultado do teste. Ao ser pressionado o botão ligado ao *ARM*, essa informação passa do *ARM* para o *FPGA* e chega ao *LED* do outro lado da placa, ligado ao *FPGA*.

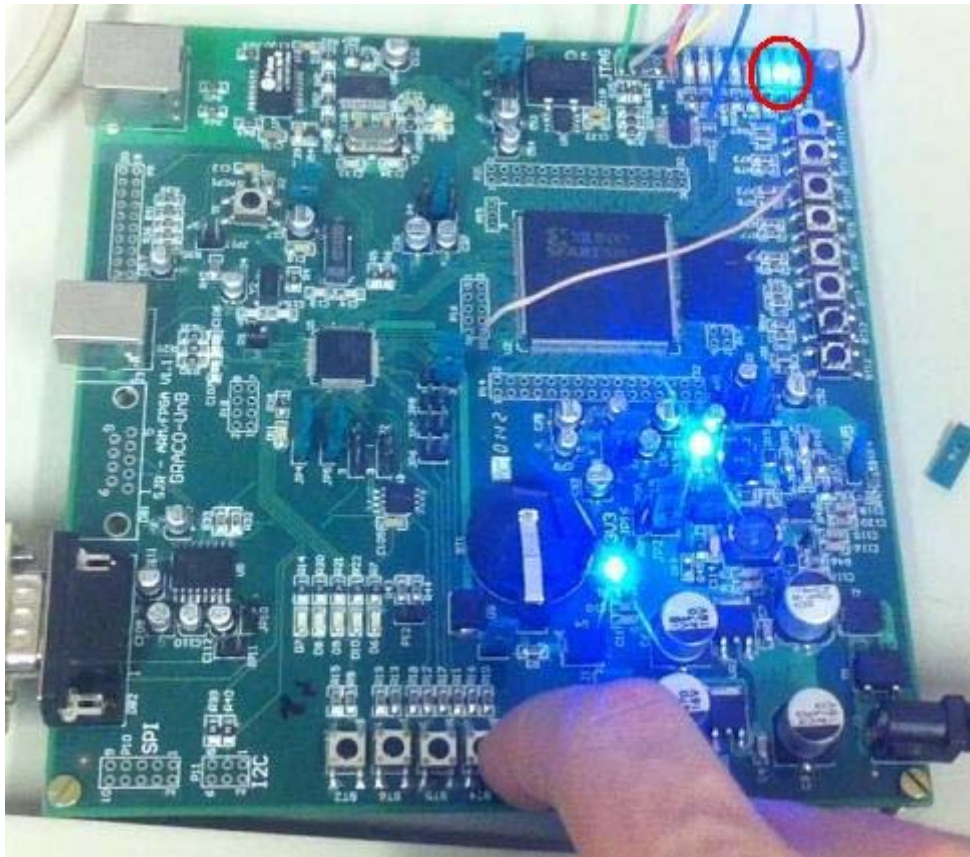



Figura 49 – Foto do teste de funcionamento do *ARM* e *FPGA* integrados**.**

5.5 CONCLUSÕES DO CAPÍTULO

A fase de testes constitui a prova de fogo para qualquer projetista. É nela que são identificadas as imperfeições, erros de projeto bem como o bom funcionamento, conforme o caso.

Obter uma placa sem necessidade de melhorias ou ajustes em sua primeira versão, em nível acadêmico, é praticamente impossível. Levando-se em consideração a complexidade da plataforma diante da capacidade da equipe, considera-se que os resultados obtidos superaram as expectativas.

Algumas falhas de projeto eram, de certa forma, até mesmo esperadas que fossem descobertas durante a fase de testes. Considerando a quantidade de circuitos implementados e a quantidade de erros fundamentais identificados (apenas 3, vide sessão 6.3) é possível afirmar que o projeto obteve sucesso no atingimento de seus objetivos. Os erros foram facilmente corrigidos e, com isso, prontamente estabelecido o pleno funcionamento da plataforma.

Destaca-se, mais uma vez, que o objetivo dos testes não era demonstrar o pleno potencial da plataforma e as incontáveis possibilidades de utilização dela no controle de processos. Os testes dedicaram-se tão somente a demonstrar que as principais funções da plataforma estão em funcionamento e podem ser utilizadas conforme a necessidade do projetista.

Considera-se que o mais relevante e condizente com o objetivo do projeto como um todo foi obter, já na primeira versão, uma plataforma operacional com os dois núcleos operando em regime de normalidade sem qualquer sinal de anomalia ou mal funcionamento identificado.

CAPÍTULO 6
CONCLUSÕES

6.1 CONSIDERAÇÕES GERAIS

O objetivo primário deste trabalho consistiu em projetar, desenvolver e implementar uma plataforma híbrida reconfigurável voltada para o público acadêmico de automação e controle ou desenvolvedores de novos produtos ligados à automação.

Adicionalmente, também foi proposta a elaboração e aplicada uma metodologia de desenvolvimento e testes para o sistema. Essa metodologia irá auxiliar novos desenvolvimentos dentro e fora do GRACO-UnB.

A plataforma desenvolvida cumpre seu papel de independência de seus dois núcleos (*ARM7* e *FPGA*), mas também possibilita a interconexão entre eles a depender da necessidade do desenvolvedor.

Pelo aspecto técnico, o *ARM7* se mostrou bastante satisfatório ao fim desejado. Ele possui várias opções de interface e comunicação externa, apresenta alto desempenho e preço que é equivalente a alguns microcontroladores de 8-bits (o *ARM* escolhido possui 32). A maioria das propriedades do *ARM* foram exploradas e utilizadas. Optou-se por concentrar a maior parte das funcionalidades da plataforma no *ARM* de modo que o componente de *hardware* reconfigurável fique praticamente a disposição do utilizador da plataforma.

A escolha do *ARM7* remete à tradição de um componente que já vem sendo utilizado largamente em equipamentos de utilização em massa e a comprovação, frente ao mercado, da combinação favorável entre qualidade, desempenho e preço.

A escolha do *FPGA* incorpora ao projeto uma tecnologia mais recente possibilitando uma solução baseada em *hardware*. De fato, sistemas

embarcados estão cada dia mais dependentes de tecnologias reconfiguráveis e como exposto, atualmente é impensável a utilização de sistemas embarcados sem considerar a utilização de *FPGAs*. Além do que eles são vistos atualmente como *a solução* mais tangível para se conseguir trazer os supercomputadores ao cotidiano das pessoas e ainda representam a solução ecologicamente mais viável pelo menor consumo energético em equivalência de performance se comparados aos *GPPs*.

A qualidade da plataforma foi primada desde a elaboração dos circuitos utilizados passando pelas funcionalidades implementadas, seleção de componentes adequados, materiais apropriados, empresas capacitadas para a fabricação da placa e a exaustão na fase de testes.

A robustez pôde ser verificada pela adequada seleção de circuitos e componentes, todos, seguindo rigorosamente as recomendações dos fabricantes e, algumas vezes, superdimensionados. Na fase de testes, observou-se que as fontes e capacitores, por exemplo, funcionaram bem abaixo dos limites de operação, o que significa segurança para o projetista durante o uso da plataforma.

As diversas interfaces disponibilizadas na plataforma oferecem muitas possibilidades de interconexão ou mesmo utilização, características essenciais para sistemas embarcados. Além das diversas vias de comunicação, sete pinos não utilizados dos *ARM* e setenta e oito do *FPGA* foram disponibilizados em barras de pinos para que o desenvolvedor possa fazer uso deles da maneira como queira. Esse conjunto garante a flexibilidade da plataforma aos mais diversos projetos de controle e automação.

Buscou-se em todo tempo compatibilizar a qualidade desejada com um baixo custo de produção. Para isso, foram selecionados, entre as opções com garantia de qualidade, tradição e preocupação ambiental disponíveis no mercado, aqueles componentes que apresentaram o menor custo. Por outro lado, a placa de circuito impresso foi elaborada para ser fabricada em quatro camadas que, sem dúvida, encarece o projeto, mas por outro lado favorece a blindagem eletrônica da placa e ajuda na eliminação de ruídos. Assim, as escolhas feitas não tiveram como objetivo final o preço mínimo, mas sim encontrar, dentro dos limites de projeto, uma solução de baixo custo.

Todos os componentes e subsistemas previstos no início do projeto foram passiveis de implementação e, ao final, a plataforma conta com os principais mecanismos e funcionalidades necessários ao desenvolvimento de um projeto de controle e automação que envolva gestão e processamento de sinais.

As metodologias de projeto e de testes usadas poderão servir de subsidio para desenvolvimento de outros tipos de plataformas com características de

reconfigurabilidade ou não. Elas foram desenvolvidas para serem aplicadas de maneira geral, em áreas envolvendo a mecatrônica, como automação predial/residencial (domótica), computação ubíqua, visão computacional, controle aplicado à robótica, ou qualquer outro sistema que envolva a microeletrônica ligada a sistemas de controle.

O projeto cumpriu seu papel, tanto pela obtenção do produto desejado como em termos acadêmicos pois serviu de laboratório de crescimento e amadurecimento de todos aqueles envolvidos em sua formulação. A plataforma desenvolvida apresenta funcionalidades e propriedades únicas e importantes que não são encontradas na maioria dos kits de desenvolvimento disponíveis no mercado. Isso faz com que ela tenha um mercado potencial no curto e médio prazo (devido à rápida evolução tecnológica).

6.2 COMENTÁRIOS A RESPEITO DOS OBJETIVOS ESPECÍFICOS DO TRABALHO

Além do objetivo geral do projeto, foram apresentadas na sessão 1.2.1 objetivos específicos que fazem parte do processo de desenvolvimento do todo. Cada um deles foi avaliado quanto ao seu cumprimento ou não.

Foi proposto selecionar dois núcleos de processamento para compor a base da plataforma de processamento de sinais. É de se notar que este objetivo foi integralmente cumprido vez que a plataforma conta com dois núcleos de processamento, sendo um deles um *ARM7* (LPC2146) e o outro um *FPGA* (XC3S500E).

Foi proposto selecionar circuitos periféricos para compor e dar suporte a cada um dos núcleos selecionados. Este objetivo foi integralmente cumprido por meio de um levantamento de necessidades básicas dos núcleos (*ARM* e *FPGA*) em conjunto com projetos de automação padrão que podem vir a ser implementados utilizando a plataforma. Como resultado, foram elaborados os seguintes módulos, cada um suprindo o respectivo núcleo: (a) fonte de alimentação, (b) memória externa, (c) interfaces com o usuário e (d) o de comunicação utilizando os mais diversos protocolos.

Foi proposto elaborar circuitos eletrônicos constituintes de cada núcleo separadamente. Este objetivo foi integralmente concluído por meio do design dos circuitos eletrônicos de todos os módulos constituintes da plataforma. Os testes realizados mostraram que os dois núcleos podem funcionar independente um do outro contando, inclusive, com fonte própria de alimentação.

Foi proposto elaborar mecanismos para a integração dos dois núcleos. Este objetivo foi cumprido ao utilizar diversas vias de comunicação entre os dois núcleos. Alguns protocolos, como o *Serial*, I^2C, e *SPI*, foram utilizados além de deixar alguns pinos disponíveis tanto no processador ARM quanto no FPGA para que possam ser interconectados da forma que melhor atender às necessidades do projetista. Os testes realizados mostraram que os dois núcleos podem ser integrados por meio das vias de comunicação direta entre eles. A comunicação pode ser feita nos dois sentidos, *ARM* para *FPGA* e *FPGA* para *ARM*, de maneira direta e objetiva, sem qualquer interface especial. Deve ser observado apenas a correspondência entre as tensões de referência utilizadas no *ARM* e no *bank* a ser usado no *FPGA* de modo a compatibilizar os níveis de voltagem da comunicação. Com isso, garante-se que eles possam se comunicar e trabalhar em conjunto conforme a necessidade do programador.

Foi proposto realizar a seleção dos componentes, tendo como referência os requisitos de qualidade, flexibilidade, baixo custo e compromisso ambiental. Este objetivo foi cumprido pois, na medida do possível, buscou-se utilizar componentes de alto desempenho, que podem ser facilmente encontrados para compra a um custo justo e que atendem às expectativas do projeto. Além disso, foram analisados critérios de certificação RoHS, perenidade de produção, facilidade de aquisição e confiabilidade de fornecedores para que seja possível dar continuidade da evolução do sistema.

Foi proposto elaborar o *layout* da placa de circuito impresso. Este objetivo foi integralmente cumprido conforme mostrado no APÊNDICE B. Com a verificação de bom funcionamento da plataforma, constata-se o *layout* desenvolvido mostra-se satisfatório à primeira versão da plataforma. Contudo esse *layout* pode ser melhorado em diversos aspectos em uma segunda versão conforme sugerido na sessão 6.3, a seguir.

Foi proposto ainda confeccionar a placa de circuito impresso conforme *layout* desenvolvido. Isso foi feito por meio da contratação da empresa *Print Circuits Eletronica LTDA* na data 17/11/2011, para a impressão da placa conforme projeto fornecido à empresa. Quatro placas foram entregues em janeiro de 2012, acompanhadas de fotolitos e demais documentos de fabricação.

Foi proposto também montar os componentes na placa de circuito impresso. Assim como para a confecção da PCI, foi contratada a empresa *Equitronic Equipamentos Eletrônicos Ltda*, em São Paulo, para fazer essa montagem. Esse processo não foi realizado pela equipe por falta de equipamentos e infraestrutura necessária para manusear e soldar componentes *SMD*. A plataforma foi entregue montada no início de fevereiro de 2012.

Foi proposto executar testes de *hardware* e avaliação de desempenho. Parte dessa etapa é realizada durante a fabricação da placa a qual já é entregue

testada, conforme descrito na sessão 5.1. Outra parte foi realizada em bancada de experimentação após a montagem dos componentes. Esses últimos restringiram-se a verificar o adequado funcionamento de *LEDs*, botões, reguladores de tensão, interfaces de comunicação e dispositivos essenciais. Os núcleos de processamento, por exemplo, foram testados por meio de execução de *software* e verificação de normalidade.

Na última etapa de avaliação, foi proposto executar testes de *software* e verificação de desempenho. Os testes foram feitos e comprovaram que a plataforma encontra-se operacional conforme planejado. Ela pode ser utilizada como se fosse duas absolutamente independentes ou totalmente integradas, conforme a necessidade do projeto.

O núcleo reconfigurável traz uma abordagem mais ecológica à plataforma por apresentar melhores resultados na relação desempenho por energia consumida. Já o *ARM7* dá uma roupagem mais econômica ao sistema por ganhar do *FPGA* na relação desempenho por dolar investido. Assim, a união das duas representa possibilitar ao desenvolvedor extrair o de melhor de cada subsistema para elaborar os mais diversos projetos de automação ubíqua.

6.3 SUGESTÕES PARA TRABALHOS FUTUROS

Futuros trabalhos podem ser desenvolvidos tendo como referência esta plataforma reconfigurável. As possibilidades de elaboração de novos sistemas e soluções utilizando apenas a plataforma ora apresentada são incontáveis.

Ademais, novas plataformas podem ser elaboradas, utilizando novos componentes, processadores com maior capacidade (que surgem a cada dia e cada vez mais baratos), *FPGAs* mais avançadas tecnologicamente, aumento da capacidade de memória externa de cada unidade de processamento, inserção de novas interfaces, entre outras características.

Caso seja feita a opção de promover melhorias no presente projeto, algumas observações são apresentadas de modo a direcionar o início de trabalhos futuros. Destaca-se que apenas as alterações II.b, II.c e II.p são essenciais ao funcionamento da plataforma, as demais são apenas acessórias.

I. Melhorias nos circuitos relacionados ao módulo *ARM*:

   a. Verificar a necessidade de atualizar o processador *ARM*.
   b. Adicionar uma chave de *liga/desliga* logo após o conector de alimentação.

c. Limitar a entrada de alimentação da bateria em 3,3V e não a 3,6 conforme manual de utilização do *ARM*. Para isso pode ser utilizado um diodo *zener* logo após a bateria.
d. Sequenciar a numeração de botões de 1 a 5 de forma a se apresentarem de maneira mais intuitiva. Reposicioná-los, na PCI, em ordem crescente.
e. Compatibilizar a numeração dos botões com uma sequência crescente dentro de um só PORT no microcontrolador. Na versão 1.1 o BT2 está no *Port 0* enquanto que o restante está no *Port 1*.
f. Sequenciar a numeração dos *LEDs* de 1 a 5 de forma a se apresentarem de maneira mais intuitiva.
g. Reposicionar, na PCI, os *LEDs* em ordem crescente de numeração e posicioná-los logo acima dos botões, fazendo coincidir a numeração de ambos. De preferência, utilizar uma nomenclatura mais intuitiva, como: *DA1* até *DA5*.
h. Verificar a necessidade de aumentar a capacidade da memória externa.
i. Compartilhar com o FPGA os pinos do ARM destinados à comunicação via protocolos, como I^2C, SPI, RS232, JTAG, etc.
j. Inserir a nomenclatura da *ISP/IAP* no terminal de saída P12. Adicionar um botão ao lado do terminal P12 com a mesma função para facultar ao usuário qual utilizar. Um botão pode facilitar os inúmeros processos de regravação durante a fase de desenvolvimento.
k. Aproximar o terminal P12, e o botão adicionado, do conector *COM1* por estarem diretamente relacionados.
l. Posicionar o terminal de saída da comunicação *ISP* na borda da placa. Utilizar barramento 3x2.
m. Inserir a nomenclatura da comunicação *JTAG* no terminal de saída P9. Compatibilizar a pinagem do conector P9, atualmente dedicado à comunicação *JTAG*, com o padrão utilizado pelas plataformas de programação via cabo *USB* da *Xilinx*. Vide sugestão na Figura 50.

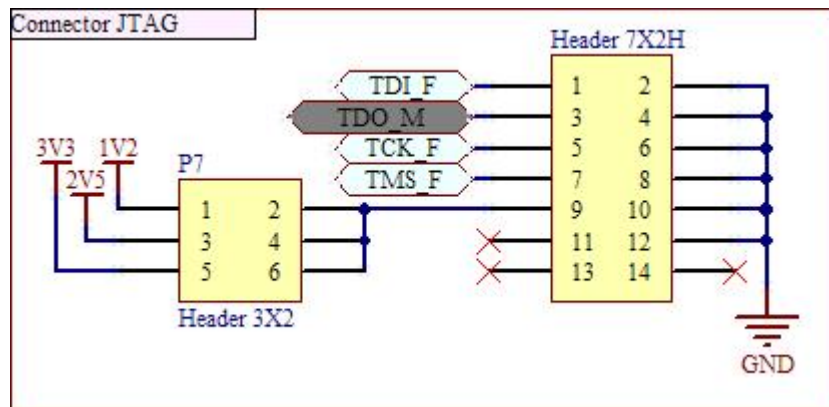



Figura 50 – Conectores padrão Xilinx para comunicação *JTAG***.**

n. Modificar os terminais de comunicação *Serial* e *SPI* conforme o padrão utilizado pela *plataforma Xilinx USB*.
o. Alterar a nomenclatura, no projeto esquemático e na PCI, dos pinos 3 e 5 no *ARM7* - de *RTXCx* para *RTCXx*.
p. Aumentar a quantidade de pinos disponibilizados ao desenvolvedor, mesmo que compartilhados com outras funções.
q. Posicionar o conector P18 na borda da placa de modo a facilitar a utilização dos pinos.
r. Tornar o terminal DB9-1 como o principal para a gravação.
s. Adicionar mais um CI *MAX3232* e fazer com que tanto o terminal DB9_1 como o DB9_2 funcionem no modo *full duplex* sem a necessidade de *jumpers*.

II. Melhorias nos circuitos relacionados ao módulo *FPGA*:

a. Verificar a necessidade de atualizar o *FPGA.*
b. Modificar o circuito de alimentação do *FPGA* pois o *mosfet*, Q1 (SI2323), está com os pinos 2 e 3 invertidos (fonte e dreno).
c. Dedicar os pinos P81(M2), P84 (M1), P86 (M0) e P206 (HSWAP) à configuração inicial da plataforma, conforme indicado no manual do fabricante do *FPGA*[31]. Disponibilizá-los em barra de pinos para que o usuário consiga fazer a configuração inicial do *FPGA* de maneira simplificada. A utilização desses 3 pinos é essencial para a programação e funcionamento do *FPGA*. A Figura 51 traz o circuito sugerido para a montagem.

[31] Disponível em: http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf.

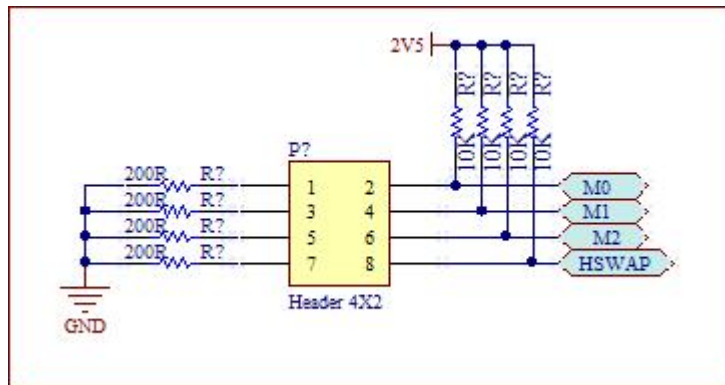



Figura 51 – Modos de seleção de inicialização do *FPGA*.

d. Atualizar os valores dos capacitores C109, C110 e C112 para 0.1uF/50V. O *footprint* está correto, apenas o valores estão desatualizados. Na montagem da plataforma foram utilizados os capacitores AVE104M50B12T-F.
e. Utilizar todos os pinos do FPGA. Foi esquecido de inserir um bloco de pinos no projeto esquemático e, por consequência, na PCI.
f. Identificar, na PCI, a polaridade dos capacitores C114 e C126.
g. Associar o CI *ICS551MLFT*, que está especificado na posição *CLK_DIST1*, ao componente U9, no *silk* da placa.
h. Fazer referência, ao lado de cada seletor de alimentação do *FPGA* (P4, P5, P6 e P7), ao *Bank* que ele está relacionado. Por exemplo, P4=>B0, representando que o seletor P4 diz respeito à alimentação utilizada no *bank 0*, e assim por diante: P5=>B1; P6=>B3 e P7=>B2.
i. Sequenciar os seletores de alimentação dos *banks* em ordem crescente, de forma a coincidir com a ordem crescente dos próprios *banks*. Agregá-los para facilitar a identificação e localização.
j. Sequenciar em ordem crescente a numeração dos *LEDs* da interface de saída do *FPGA*, de preferência com uma nomenclatura mais intuitiva, como: DF1 a DF7.
k. Reposicionar, na PCI, os *LEDs* em ordem crescente de numeração e posicioná-los logo acima dos botões, fazendo coincidir a numeração de ambos.
l. Adicionar *jumper* em todos os pinos compartilhados entre *ARM* e *FPGA* para favorecer a completa independência das unidades de processamento. Utilizar uma única barra de pinos dupla para isso. Fazer coincidir a numeração crescente dos pinos do *ARM*, do *FPGA* e da *barra de pinos*.
m. Clarificar, no *silk* da PCI, que os conectores P14, P16, P17 e P21 são pinos para livre utilização pelo programador.

n. Substituir os resistores em série com os *LEDs* indicativos de funcionamento das fontes de alimentação (3V3, 3V3A, 2V5 e 1V2). Esses *LEDs* podem drenar muita corrente e sobrecarregar os reguladores de tensão. Utilizar valores na faixa de 1KΩ.
o. Substituir o conector P8, atualmente dedicado à comunicação *JTAG*, por um 7X2 conforme padrão utilizado pelas plataformas de programação via cabo *USB* da *Xilinx*. Vide Figura 50.
p. Alterar o circuito referente à comunicação entre o *FPGA* e a memória via *JTAG*, conforme explicado no item 5.2.4. Para tanto, basta conduzir o pino P207 (TDI) do *FPGA*, passando por um resistor de 68Ω, até o pino TDI no conector de borda. Conduzir o pino P157 (TDO) do *FPGA*, direto ao pino 4 (TDI) na memória *PROM*.

REFERÊNCIAS BIBLIOGRÁFICAS

ACAR, M., “Mechatronics challenge for the higher education world”. **IEEE transactions on Components, Packing, and Manufacturing Technology**. Vol. 20, no. 1, pp. 14-20. 1997.

ADAMOWSKI, J. C. E FURUKAWA, C. M., “Mecatrônica: Uma abordagem voltada à automação industrial”. **Mecatrônica Atual**. Ed. Saber. No. 1. Pág. 08-11. Out-Nov. 2001.

AIELLO, M.; DUSTDAR, S. Are our homes ready for services? A domotic infrastructure based on the Web service stack. **Pervasive and Mobile Computing**, v. 4, n. 4, p. 506-525, ago. 2008.

AJAY KUMAR, V. **Control of PCB track impedance by adequate buffering, proper termination, careful routing strategies establishes signal quality in microprocessor based system**Electromagnetic Interference and Compatibility, 1995., International Conference on. **Anais**...Dezembro. 1995

ATUAL, M. Automação industrial de processos e manufatura. **São Paulo-Editora Saber LTDA–2010-http://www. mecatronicaatual. com. br**, [S.d.].

AUGUSTIN, I. *et al.* ISAM, joining context-awareness and mobility to building pervasive applications. **Architecture**, v. 1, p. 2, 2004.

BARTIC, T. A. *et al.* **Highly scalable network on chip for reconfigurable systems**System-on-Chip, 2003. Proceedings. International Symposium on. **Anais**...nov. 2003

BONINO, D.; CORNO, F. Modeling, simulation and emulation of Intelligent Domotic Environments. **Automation in Construction**, v. 20, n. 7, p. 967-981, nov. 2011.

BRADLEY, D. Mechatronics – More questions than answers. **Mechatronics**, v. 20, n. 8, p. 827-841, dez. 2010.

BUCHANAN, M. **Ubiquity: the science of history ... or why the world issimpler than we think**. 1st American ed ed. New York: Crown Publishers, 2001.

BURGE, S. E.; GROUT, I. A.; DOREY, A. P. **Application specific integrated circuit implementation of SISO control-laws**Control, 1994. Control '94. International Conference on. **Anais**...mar. 1994

CANZIAN, E. **Comunicação Serial - RS232**, [S.d.]. Disponível em: <http://www.professores.aedb.br/arlei/AEDB/Arquivos/rs232.pdf>

CASTELLS, M.; MAJER, R. V.; GERHARDT, K. B. **A sociedade em rede**. [S.l.] Paz e terra São Paulo, 2000. v. 3

CHAGAS, R. A. J. **CONSTRUÇÃO DE UM SISTEMA PARA AUTOMAÇÃO PREDIAL E RESIDENCIAL UTILIZANDO O PROTOCOLO BACNET/IP**. Brasília: Universidade de Brasília, jul. 2008.

CHEN, R.-X.; CHEN, L.-G.; CHEN, L. System design consideration for digital wheelchair controller. **Industrial Electronics, IEEE Transactions on**, v. 47, n. 4, p. 898 -907, ago. 2000.

COMPTON, K.; HAUCK, S. Reconfigurable computing: a survey of systems and *software*. **ACM Computing Surveys (csuR)**, v. 34, n. 2, p. 171–210, 2002.

CORIC, S. *et al.* Parallel-beam backprojection: an FPGA implementation optimized for medical imaging. **The Journal of VLSI Signal Processing**, v. 39, n. 3, p. 295–311, 2005.

CORREIA, A. P. **UM PROJETO DE CONTROLE DE MOVIMENTAÇÃO VEICULAR PROJETADO EM UM PROCESSADOR EMBARCADO EM FPGA COM AMBIENTE DE SIMULAÇÃO USANDO INSTRUMENTAÇÃO VIRTUAL**. Brasilia: Universidade de Brasilia, jun. 2007.

COSTA, A. M. N. DA. Revoluções tecnológicas e transformações subjetivas. **Psicologia: teoria e pesquisa**, v. 18, n. 2, p. 193–202, 2002.

DAGNINO, A. **Coordination of *hardware* manufacturing and *software* development lifecycles for integrated systems development**Systems, Man, and Cybernetics, 2001 IEEE International Conference on. **Anais**...2001

DEBES, E. *et al.* **Characterizing multimedia kernels on general-purpose processors**Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on. **Anais**...2002

DIDO, J. *et al.* **A flexible floating-point format for optimizing data-paths and operators in FPGA based DSPs**Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays. **Anais**...2002Disponível em: <http://dl.acm.org/citation.cfm?id=503056>. Acesso em: 4 jul. 2012

DIODES INCORPORATED. **DN98 ZXLD1370 PCB Layout Guidelines Introduction**, 2010. Disponível em: <http://www.diodes.com/_files/design_note_pdfs/DN98%20ZXLD130%20PCB%20layout%20guidelines%20Final.pdf>. Acesso em: 6 jul. 2012

EDWARDS, M.; GREEN, P. Run-time support for dynamically reconfigurable computing systems. **Journal of Systems Architecture**, v. 49, n. 4–6, p. 267-281, set. 2003.

FARRAHI, A. H. *et al.* **Quality of EDA CAD tools: definitions, metrics and directions**Quality Electronic Design, 2000. ISQED 2000. Proceedings. IEEE 2000 First International Symposium on. **Anais**...2000

FERNANDEZ FILHO, J. S.-P. KEDBG-JTAG: UM DEPURADOR ARMJTAG PARA O ERESI REVERSE ENGINEERING FRAMEWORK. p. 56, 2009.

GHASEMI, H. R. *et al.* **Augmenting general purpose processors for network processing**Field-Programmable Technology (FPT), 2003. Proceedings. 2003 IEEE International Conference on. **Anais**...dez. 2003

GREGSON, R. J. *et al.* **The development of application specific integrated circuits using the System Design Synthesis Tool**ASIC Seminar and Exhibit, 1989. Proceedings., Second Annual IEEE. **Anais**...set. 1989

GUAN, S.; GUO, F. **A fast and accurate positioning algorithm of PCB location hole coordinate base on the potential function**Environmental Science and Information Application Technology (ESIAT), 2010 International Conference on. **Anais**...jul. 2010

HAMADA, T. *et al.* **A Comparative Study on ASIC, FPGAs, GPUs and General Purpose Processors in the O(N^2) Gravitational N-body Simulation**Adaptive *Hardware* and Systems, 2009. AHS 2009. NASA/ESA Conference on. **Anais**...29 ago. 2009

HARTENSTEIN, R. **Why we need reconfigurable computing education: Introduction**The 1st International Workshop on Reconfigurable Computing Education (RCeducation 2006), March. **Anais**...2006

HARTENSTEIN, R. Basics of Reconfigurable Computing. **Designing embedded processors: a low power perspective**, p. 451, 2007.

HUBNER, J. *et al.* Efficient data averaging for spin noise spectroscopy in semiconductors. **Applied Physics Letters**, v. 97, n. 19, p. 192109 - 192109-3, nov. 2010.

IBRAHIM, M. Y. **Myth and reality of mechatronics in tertiary education**Industrial Technology, 2003 IEEE International Conference on. **Anais**...dez. 2003

ITO, S. A.; CARRO, L. **A comparison of microcontrollers targeted to FPGA-based embedded applications**Integrated Circuits and Systems Design, 2000. Proceedings. 13th Symposium on. **Anais**...2000

KAZMIERKOWSKI, M. P. On-Chip Integration and Industrial Electronics[review of “Communication Architectures for Systems-on-Chip” (Ayala, J.L.; 2011) [Book News]. **Industrial Electronics Magazine, IEEE**, v. 5, n. 3, p. 85, set. 2011.

KEAN, T. **Cryptographic rights management of FPGA intellectual property cores**Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays. **Anais**...2002Disponível em: <http://dl.acm.org/citation.cfm?id=503065>. Acesso em: 4 jul. 2012

KIM, Y.; PARK, E. K.; TAK, S. Dynamically reconfigurable *hardware–software* architecture for partitioning networking functions on the SoC platform. **Journal of Systems and *Software***, v. 82, n. 10, p. 1588-1599, out. 2009.

KUNITO, G. *et al.* **Architecture for Providing Services in the Ubiquitous Computing Environment**Distributed Computing Systems Workshops, 2006. ICDCS Workshops 2006. 26th IEEE International Conference on. **Anais**...jul. 2006

MACHADO JUNIOR, M. M. PROTÓTIPO DE SOFTWARE PARA O INTERFACEAMENTO E AQUISIÇÃO DE DADOS DE UMA BALANÇA ATRAVÉS DA RS-232. 1999.

MEDEIROS JÚNIOR, E. F.; SILVA, G. A. C. Desenvolvimento de ECG com um canal e interface USB através do microcontrolador 8051. 2006.

MICROCHIP. **MCP120/130 - Microcontroller Supervisory Circuit with Open Drain Output**, [S.d.]. Disponível em: <http://ww1.microchip.com/downloads/en/devicedoc/11184d.pdf>. Acesso em: 5 jul. 2011

MICROPRESS. **Tabela Comparativa de Laminados para Circuito Impresso**. Disponível em: <http://micropress.com.br//pt/info-tecnicas/comparacao-laminados/>. Acesso em: 6 jul. 2012.

MIYAZAKI, T. **Reconfigurable systems: a survey**Design Automation Conference 1998. Proceedings of the ASP-DAC’98. Asia and South Pacific. **Anais**...1998Disponível em:

<http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=669520>. Acesso em: 4 jul. 2012

NAVEDA, J. F.; CHANG, K. C.; DU, H. C. **A New Approach to Multi-Layer PCB Routing with Short Vias**Design Automation, 1986. 23rd Conference on. **Anais**...jun. 1986

PEDROSO, L. J. **Projeto de placas de circuito impresso**, [S.d.]. Disponível em: <http://www.nabucoeletronica.com.br/files/pci.pdf>

PEREIRA, C. E.; CARRO, L. Distributed real-time embedded systems: Recent advances, future trends and their impact on manufacturing plant control. **Annual Reviews in Control**, v. 31, n. 1, p. 81-92, 2007.

PEREIRA JR, A. C.; SOUSA, F. L. DE; VLASSOV, V. UM ESTUDO SOBRE O POSICIONAMENTO ÓTIMO DE COMPONENTES ELETRÔNICOS EM PLACAS DE CIRCUITO IMPRESSO UTILIZANDO ALGORITMOS EVOLUTIVOS. [S.d.].

PIMENTEL, J. C. G.; LE-HUY, H. **A VHDL-based methodology to develop high performance servo drivers**Industry Applications Conference, 2000. Conference Record of the 2000 IEEE. **Anais**...2000

RAMMIG, F. **Visions from the IT Engine Room**, [S.d.]. Disponível em: <http://www.ifip.org/secretariat/tc_visions/tc10_visions.htm>

RENNER, F.-M. *et al.* Design methodology of application specific integrated circuits for mechatronic systems. **Microprocessors and Microsystems - Embedded *Hardware* Design**, v. 24, n. 2, p. 95-103, 2000.

RO, W. W.; GAUDIOT, J.-L. A complexity-effective microprocessor design with decoupled dispatch queues and prefetching. **Parallel Computing**, v. 35, n. 5, p. 255-268, maio. 2009.

ROSÁRIO, J. M. **Princípios de Mecatronica**. 1. ed. São Paulo: Pearson Prentice Hall, 2005.

SAINT-JEAN, N. *et al.* **Application Case Studies on HS-Scale, a MP-SOC for Embbeded Systems**Embedded Computer Systems: Architectures, Modeling and Simulation, 2007. IC-SAMOS 2007. International Conference on. **Anais**...jul. 2007

SATYANARAYANAN, M. Pervasive computing: vision andchallenges. 2001.

SHIMA, M. **The Birth, Evolution and Future of the Microprocessor**Computer and Information Technology, 2005. CIT 2005. The Fifth International Conference on. **Anais**...set. 2005

SILVESTRE, E.; BACHIEGA, P. **ESTUDO SOBRE PROCESSADOR ARM7**, 12 fev. 2007. Disponível em: <http://www.lisha.ufsc.br/teaching/sys/ine5309-2006-2/work/g4/monografia.pdf>

SOUZA, D. J. DE; LAVINIA, N. C. **Conectando o PIC**. 3. ed. [S.l.] Ed. Érica, 2002.

SRIVASTAVA, S. *et al.* **Evolution of architectural concepts and design methods of microprocessors**VLSI Design, 1998. Proceedings., 1998 Eleventh International Conference on. **Anais**...jan. 1998

SU, X. *et al.* **Enabling engineering document in mobile computing environment**World of Wireless, Mobile and Multimedia Networks, 2006. WoWMoM 2006. International Symposium on a. **Anais**...0. 2006

SUDHA, R. *et al.* **Ubiquitous Semantic Space: A context-aware and coordination middleware for Ubiquitous Computing**Communication Systems Software and Middleware, 2007. COMSWARE 2007. 2nd International Conference on. **Anais**...jan. 2007

TALLA, D.; JOHN, L. K. **Facts and myths about media processing on general-purpose processors**Information Technology: Research and Education, 2003. Proceedings. ITRE2003. International Conference on. **Anais**...ago. 2003

UTTERBACK, J. M. **Dominando a dinâmica da inovação**. [S.l.] Qualitymark, 1996.

VARGAS, R. V. **Gerenciamento de Projetos - Estabelecendo Diferenciais Competitivos**. 7. ed. [S.l.] Brasport, 2009.

VUKOSAVLJEV, S. A. *et al.* **A Generic Embedded Robot Platform for Real Time Navigation and Telepresence Abilities**Engineering of Computer Based Systems (ECBS-EERC), 2011 2nd Eastern European Regional Conference on the. **Anais**...set. 2011

WEI, H. *et al.* **Research on reconfigurable robot controller based on ARM and FPGA**Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on. **Anais**...jul. 2008

WEISER, M. The computer for the 21st century. **Scientific American**, v. 265, n. 3, p. 94–104, 1991.

WEISER, M. Hot topics-ubiquitous computing. **Computer**, v. 26, n. 10, p. 71–72, 1993.

WESTINE, P. S. Wages through the Ages: The Influence of Technology on the Standard of Living. **Technology and Society Magazine, IEEE**, v. 5, n. 3, p. 15 -18, set. 1986.

WIKIPEDIA CONTRIBUTORS. **ARM7**Wikimedia Foundation, Inc., 3 jul. 2012. (Nota técnica).

WOOD, R. J. Robotic manipulation using an open-architecture industrial arm: a pedagogical overview [Education]. **Robotics Automation Magazine, IEEE**, v. 15, n. 3, p. 17 -18, set. 2008.

XILINX. **Platform Cable USB**, 14 maio. 2008. Disponível em: <http://www.xilinx.com/support/documentation/data_sheets/ds300.pdf>. Acesso em: 2 fev. 2012

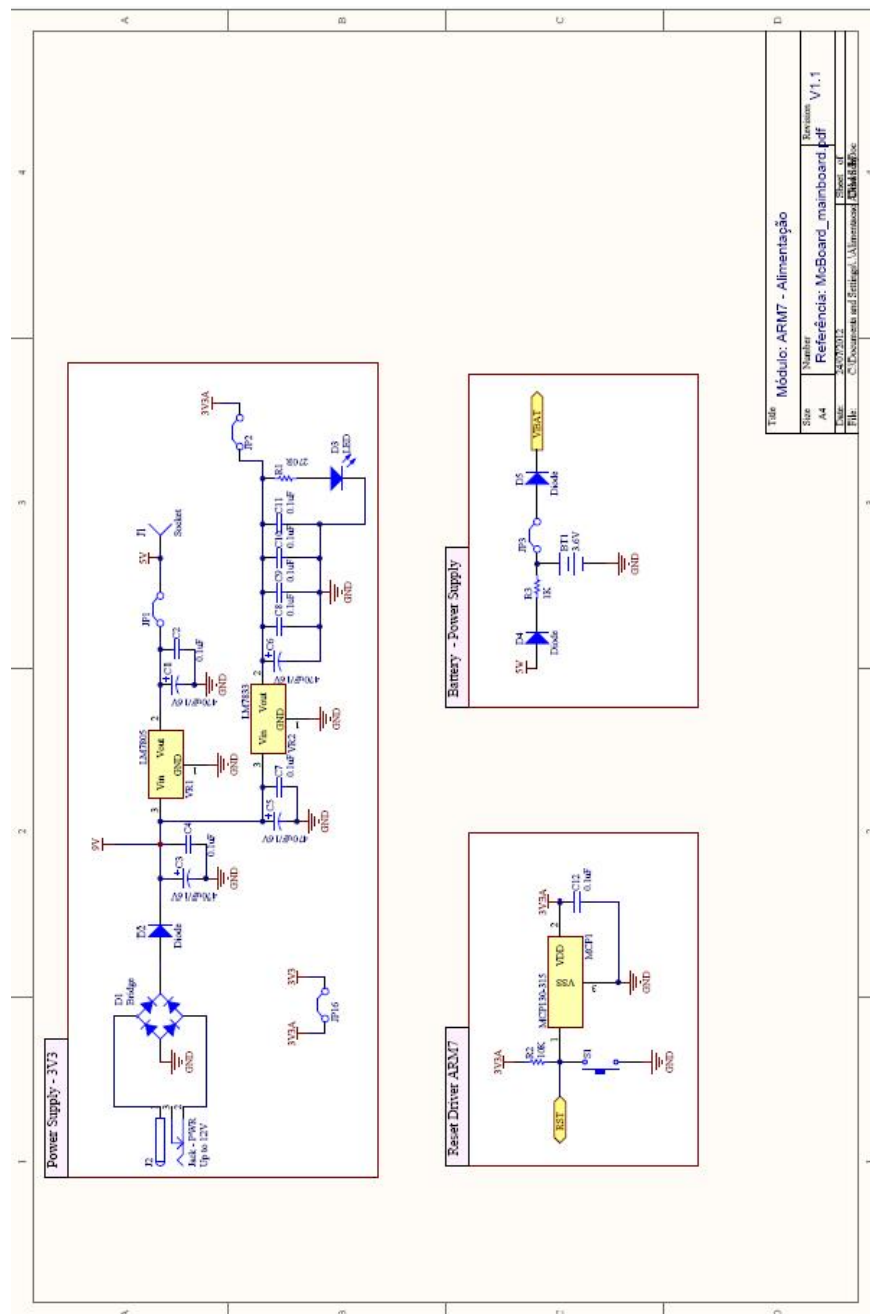
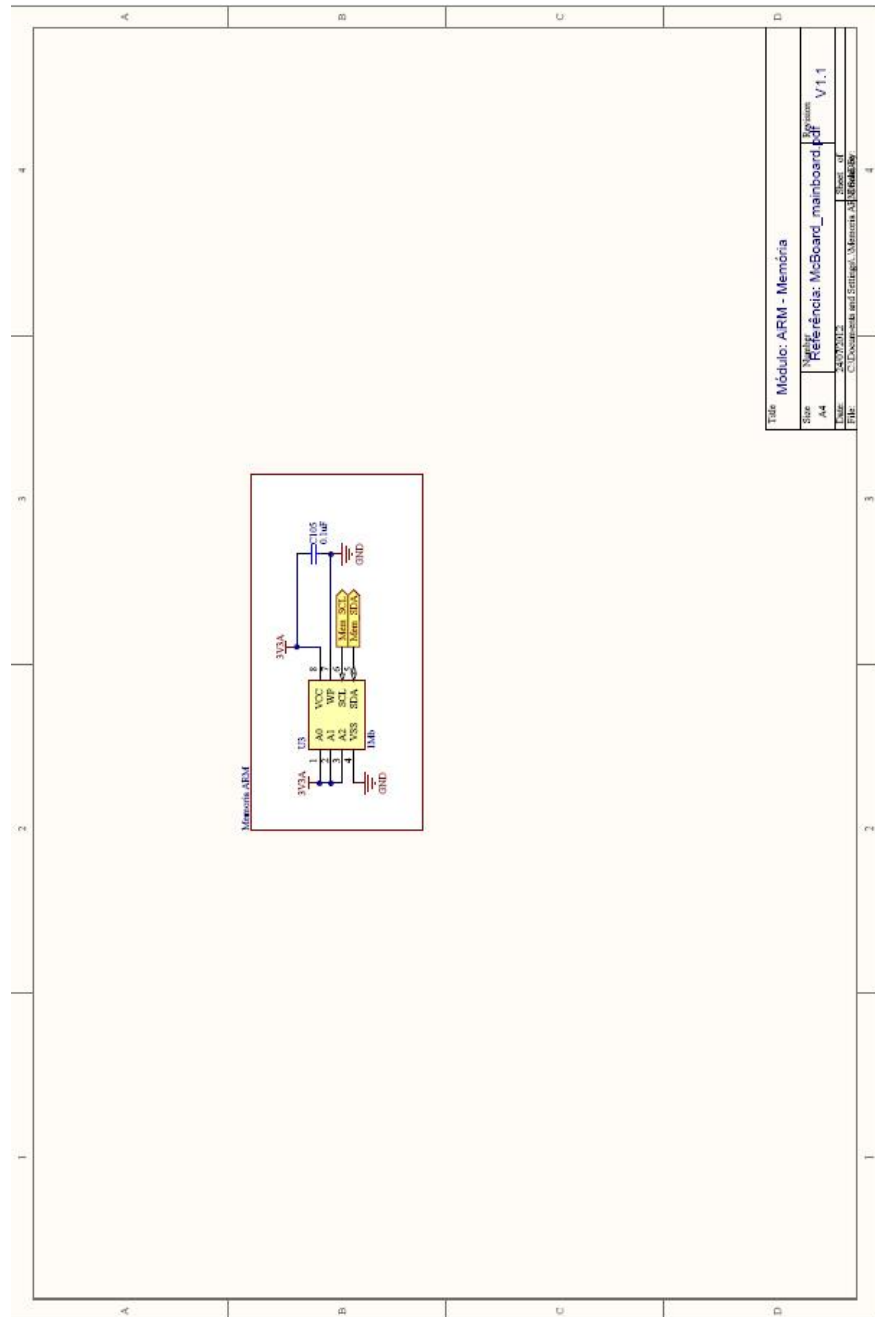
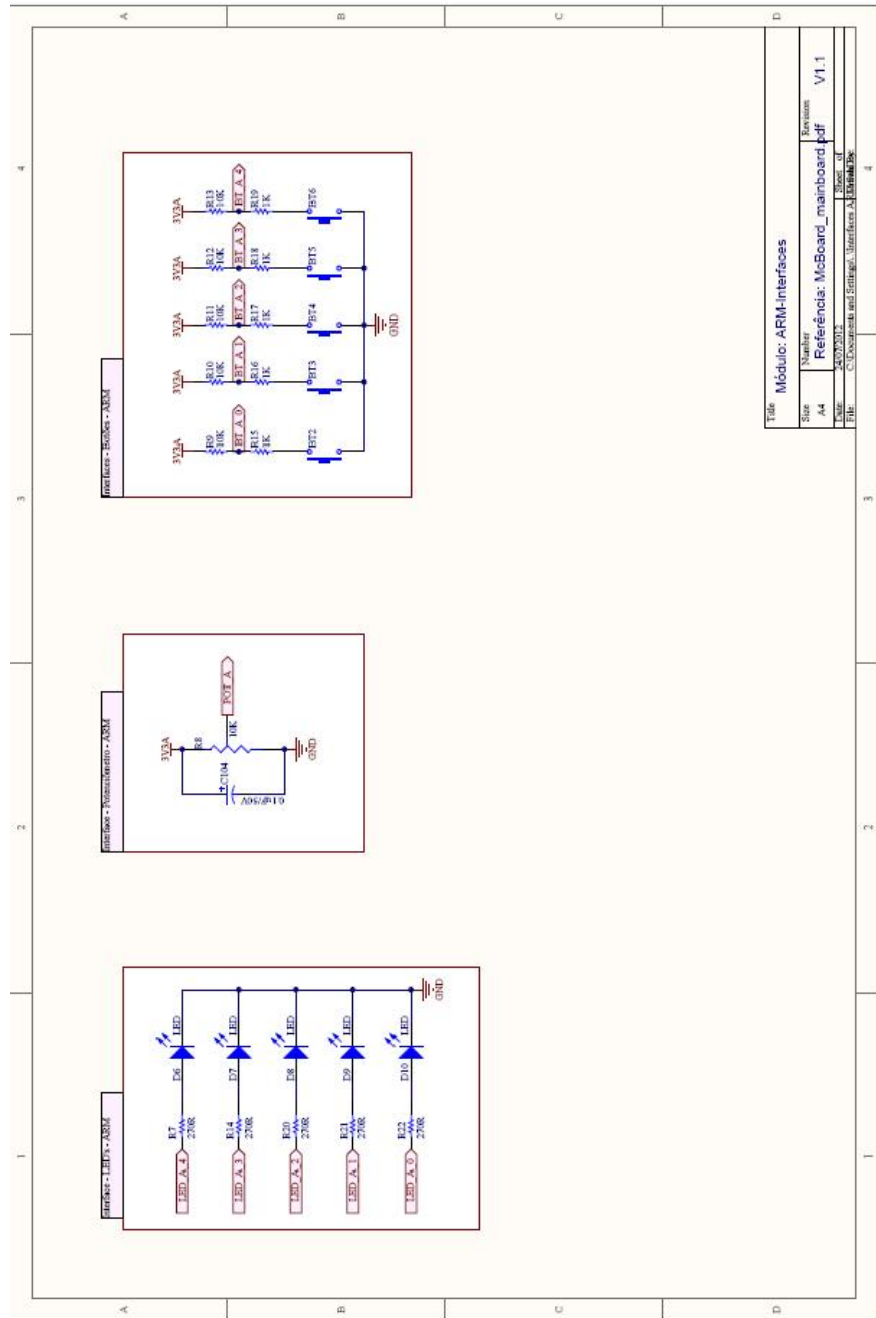
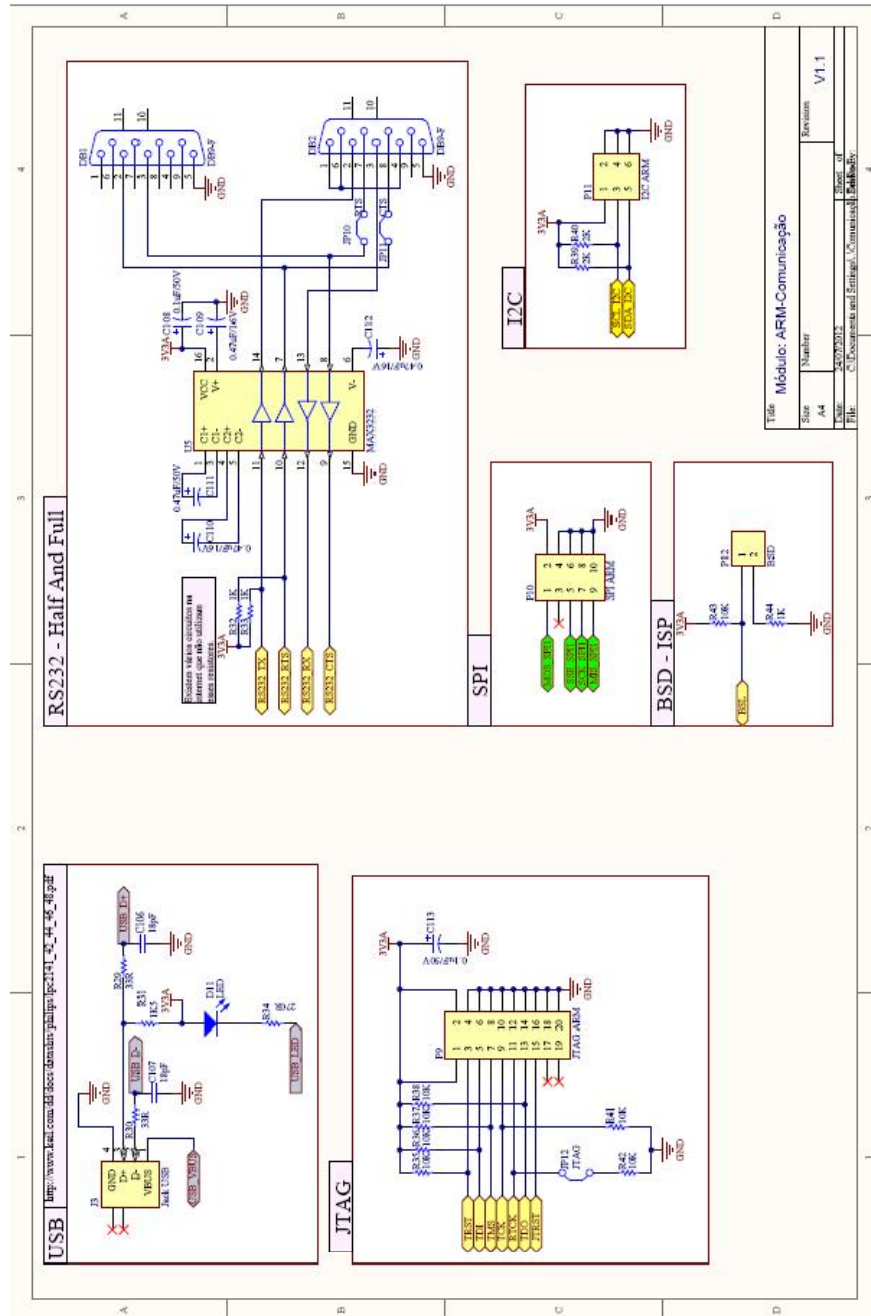
XILINX. **The 3.3V Configuration of Spartan-3 FPGAs**, 23 jun. 2008. Disponível em: <http://www.xilinx.com/support/documentation/application_notes/xapp453.pdf>. Acesso em: 1 jun. 2010

XU, J.; LI, J.; JIANG, Y. **Components Locating in PCB Fault Diagnosis Based on Infrared Thermal Imaging**Information and Computing Science, 2009. ICIC '09. Second International Conference on. **Anais**...maio. 2009

YIU, J. CHAPTER 18 - Porting Applications from the ARM7 to the Cortex-M3. *In*: **The Definitive Guide to the ARM Cortex-M3 (Second Edition)**. Oxford: Newnes, 2010. p. 283-289.

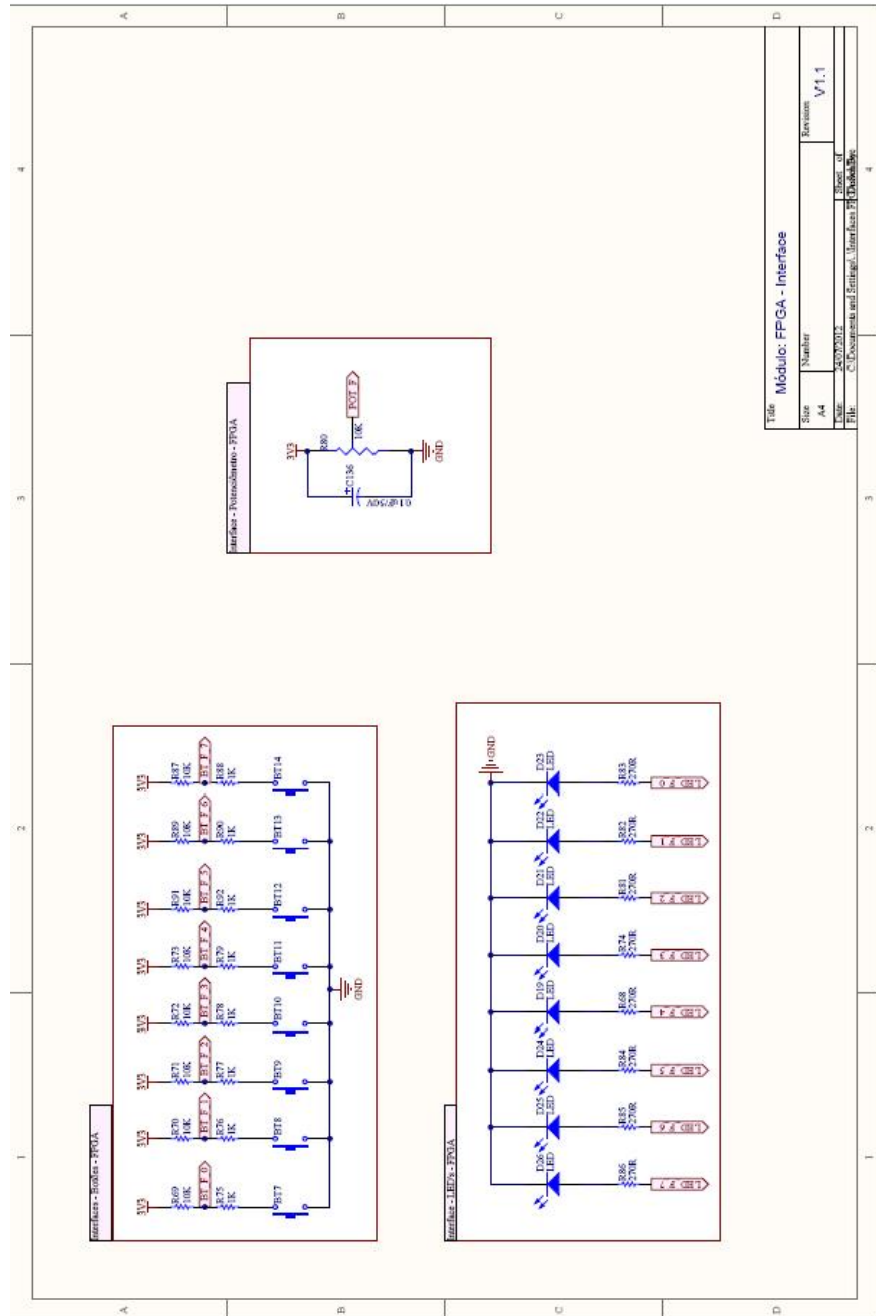
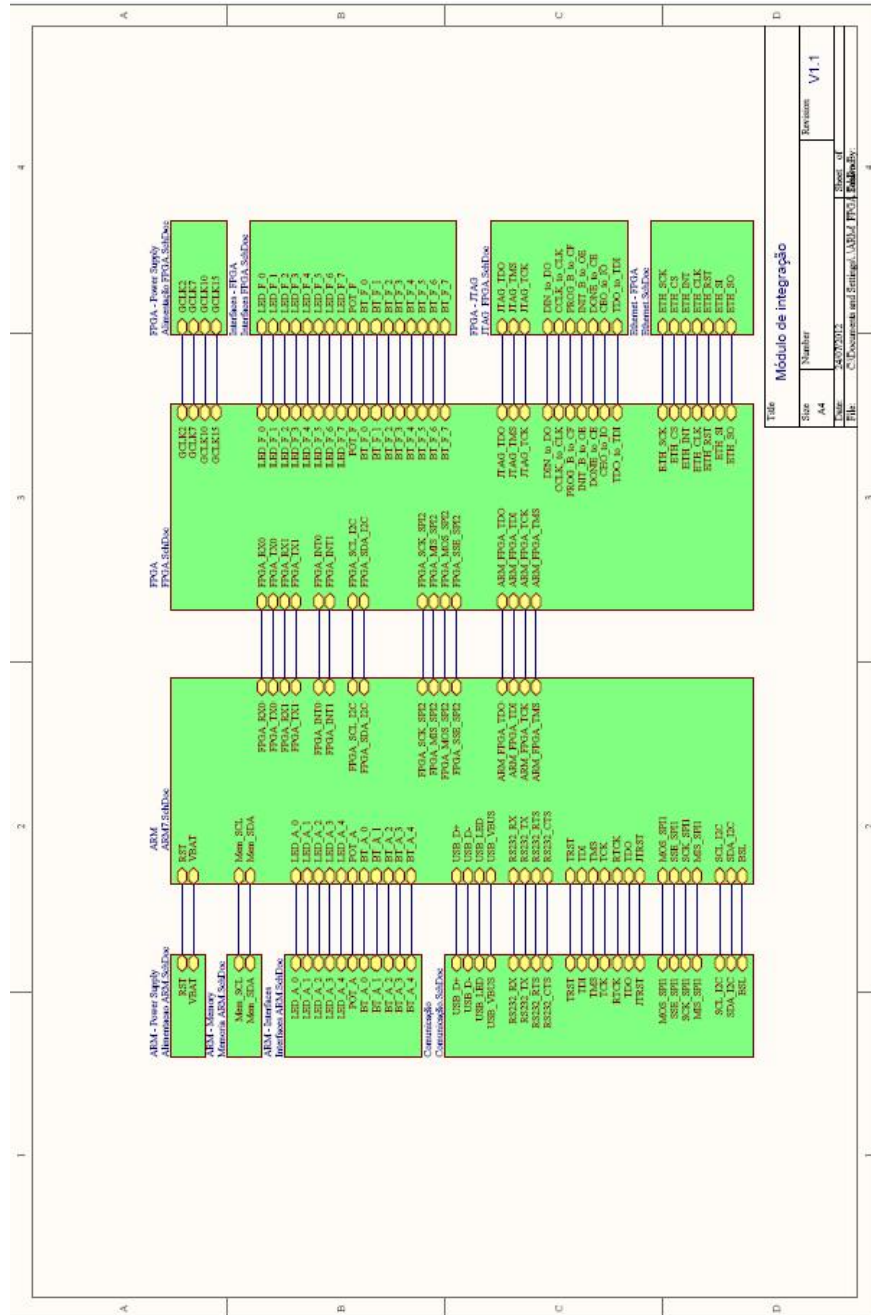
APÊNDICE A. CIRCUITOS ELETRÔNICOS (VERSÃO 1.1)

A seguir são apresentados todos os circuitos elaborados para a confecção da plataforma.

A.1 ARM – Fonte de Alimentação.

A.2 ARM – Memória

A.3 ARM - Interfaces

A.4 ARM - Communication

D.1 *ARM – CORE*

A.5 FPGA – Fonte de alimentação

A.6 FPGA – Memória – JTAG

A.7 FPGA – Interfaces

A.8 FPGA – Comunicação

A.9 FPGA – CORE

A.10 ARM - FPGA – Integração

APÊNDICE B. APRESENTAÇÃO DA PLATAFORMA

A seguir, será apresentada a disposição dos circuitos e componentes da plataforma. A Figura 52 apresenta esquematicamente a plataforma segundo, sua topologia. A Figura 53 é a foto da plataforma.

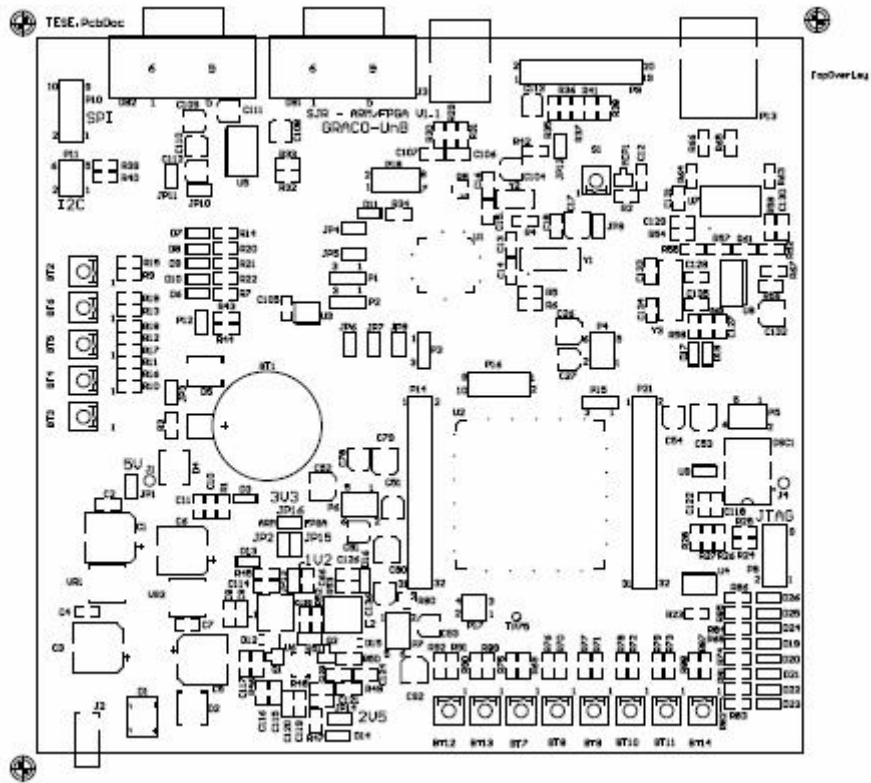



Figura 52 – Representação esquemática da PCI da plataforma reconfigurável**.**

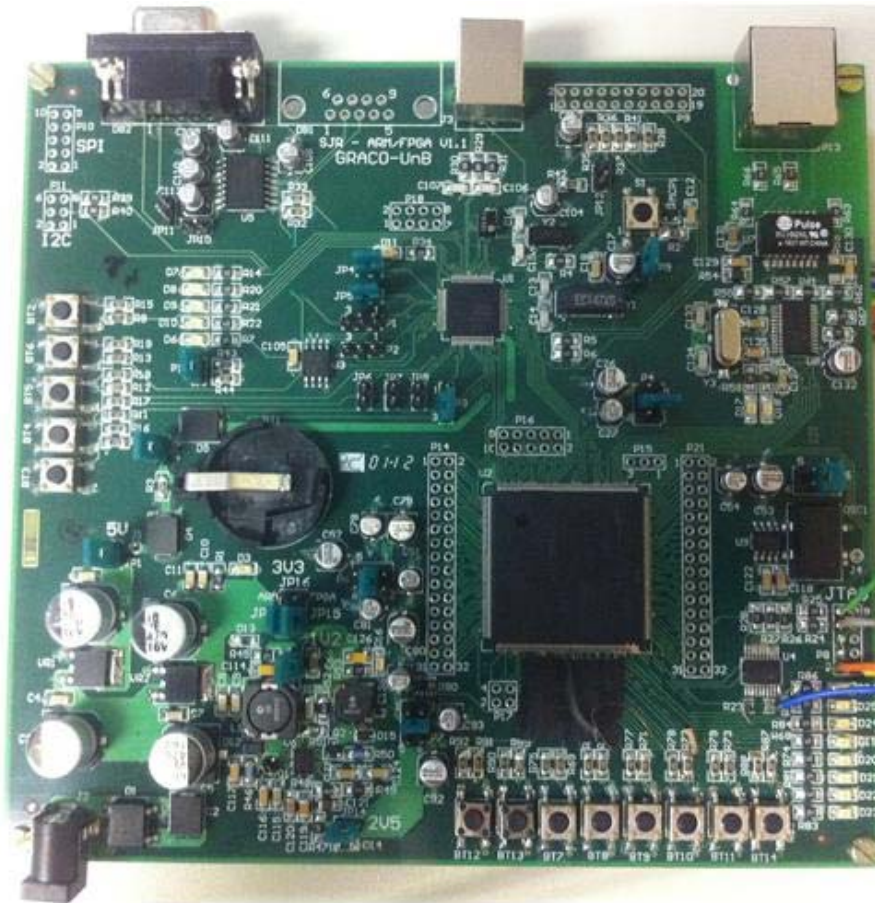



Figura 53 – Foto da plataforma reconfigurável**.**

B.1 SISTEMA DE ALIMENTAÇÃO DO ARM (9V, 5V E 3V3A)

A Figura 54 mostra o posicionamento dos principais componentes do sistema de alimentação do *ARM*.

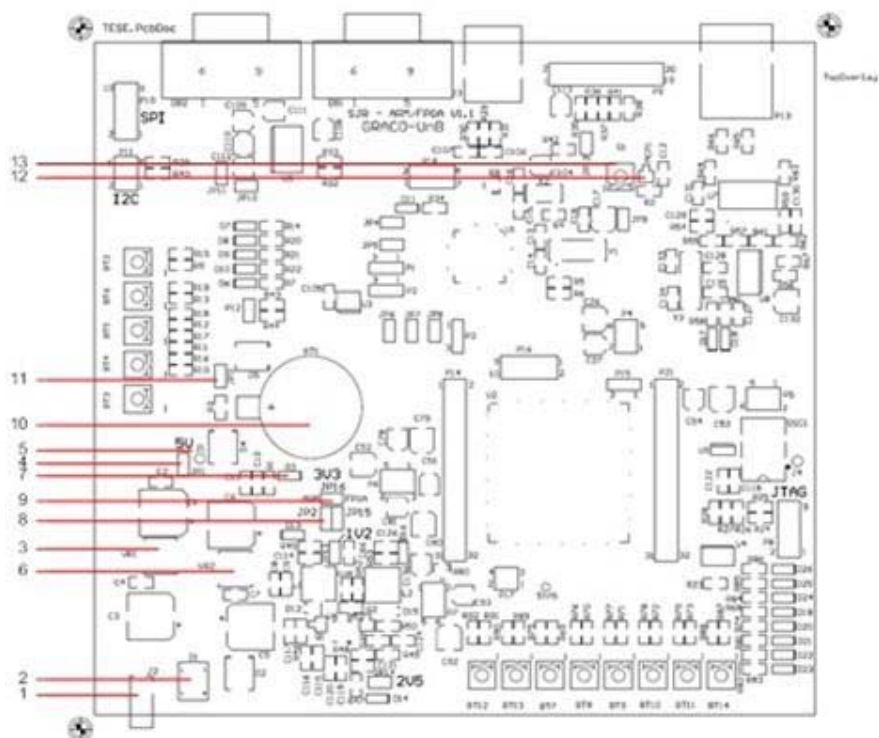



Figura 54 – Localização dos componentes do módulo de alimentação do *ARM* – *Alimentação.*

Tabela 21 – Descrição dos componentes referentes ao módulo *ARM – Alimentação*.

Item	***Nome***	***DESCRIÇÃO***	***FUNÇÃO***
1	J2	*Power Jack*	Conector de entrada de energia
2	D1	*Bridge*	Ponte de diodos para regulagem de tensão
3	VR1	*Voltage Regulator*	Regulador de tensão 5V – LM7805
4	JP1	*Jumper*	Quando desconectado, isola o circuito de suprimento de energia em nível de tensão de 5V
5	J1	*Socket*	Disponibiliza a tensão de 5V
6	VR2	*Voltage Regulator*	Regulador de tensão que disponibiliza a tensão em 3,3V – LM7833
7	D3	*LED*	Indicação visual de que o nível de tensão 3,3V está ativa, proveniente do regulador LM7833
8	JP2	*Jumper*	Quando desconectado, isola o circuito de suprimento de energia de 3,3V, proveniente do regulador LM7833, para o *ARM*

9	JP16	*Jumper*	Quando conectado, compartilha a alimentação do regulador LM7833 entre o *ARM* e o *FPGA*. Neste caso, ambos serão alimentados com a tensão 3,3V proveniente do regulador LM7833
10	BT1	*Battery*	Soquete para utilização de alimentação proveniente da bateria
11	JP3	*Jumper*	Quando desconectado, isola o suprimento de energia proveniente da bateria
12	MCP1	*Driver Reset*	Circuito integrado de gerenciamento de reinicialização do *ARM*
13	S1	*Push Button*	Botão para reinicialização do microcontrolador *ARM*

B.2 MÓDULO DE MEMÓRIA PARA O *ARM* (*ARM - MEMÓRIA*)

A Figura 55 mostra o posicionamento da memória externa ligada ao *ARM*.

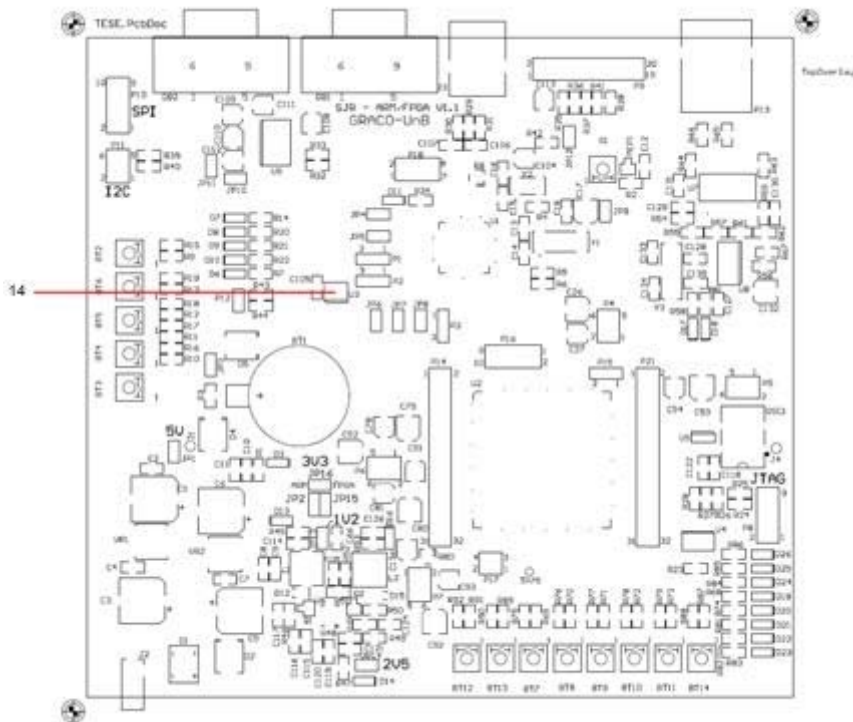



Figura 55 – Localização da memória módulo do *ARM – Memória***.**

Tabela 22 – Descrição dos componentes referentes ao módulo *ARM – Memória*.

Item	*Nome*	*DESCRIÇÃO*	*FUNÇÃO*
14	U3	*Circuito Integrado – 24LC1025*	Memória externa do *ARM*. Capacidade 1M*byte*.

B.3 MÓDULO DE INTERFACES DO ARM

A Figura 56 mostra o posicionamento dos dispositivos de interfaces (botões, *LEDs* e potenciômetro) ligados ao *ARM*.

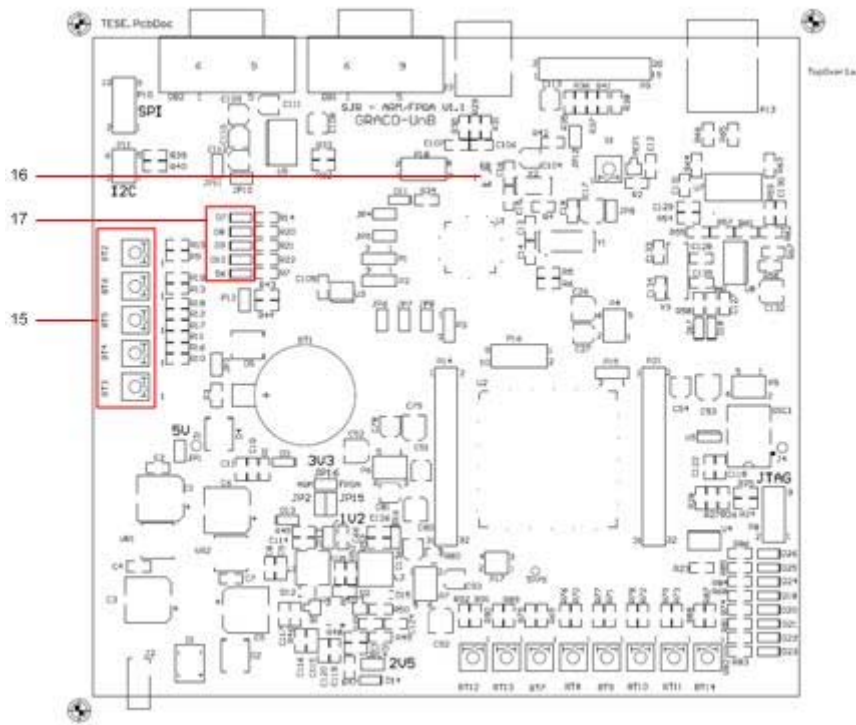



Figura 56 – Localização dos componentes do módulo de interfaces do *ARM – INTERFACES*.

Tabela 23 – Descrição dos componentes referentes ao módulo *ARM – INTERFACES*.

Item	*Nome*	*DESCRIÇÃO*	*FUNÇÃO*
15	BT2-BT6	*Push Button*	Interface de botões com o usuário
16	R8	*Potenciômetro*	Potenciômetro de 10kΩ
17	LED6-LED10	*LED*	Interface de *LED* com o usuário

B.4 SISTEMA DE COMUNICAÇAO DO ARM

A Figura 57 mostra o posicionamento dos conectores e principais componentes de comunicação ligados ao *ARM*.

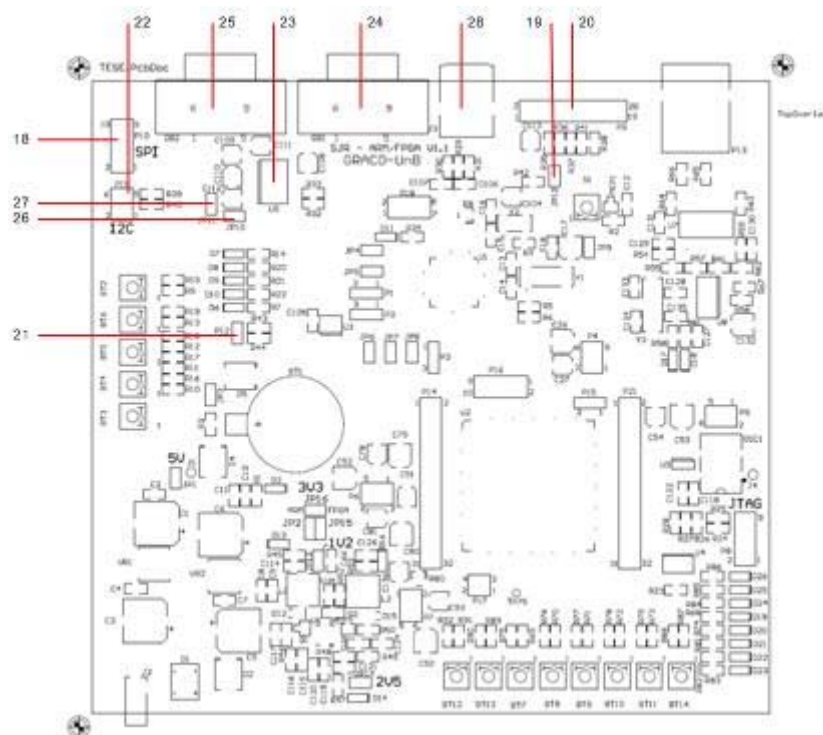



Figura 57 – Localização dos componentes do módulo *ARM-Comunicação***.**

Tabela 24 – Descrição dos componentes referentes ao módulo *ARM-Comunicação*.

Item	***Nome***	***DESCRIÇÃO***	***FUNÇÃO***
18	P10	*SPI*	Conector de saída do protocolo SPI
19	JP12	*Jumper*	Normalmente fechado. Dependendo do *ARM* a ser utilizado, pode ser que seja necessário utilizá-lo aberto
20	P9	*JTAG*	Conector de saída do protocolo *JTAG*
21	P12	*ISP/IAP*	Conector de saída do protocolo ISP/IAP
22	P11	*I^2C*	Conector de saída do protocolo I^2C
23	U5	*MAX3232*	Circuito integrado gerenciador do protocolo RS232
24	DB1	*DB9F*	Primeiro conector de saída do protocolo RS232. Utilizado apenas para a comunicação *half-duplex*

25	DB2	*DB9F*	Segundo conector de saída do protocolo RS232. Pode ser utilizado tanto como *half-duplex* como *full-duplex* dependendo da combinação dos *jumpers* JP10 e JP11
26	JP10	*Jumper*	Utilizá-lo fechado quando necessário comunicação RS232 full-duplex no conector DB2
27	JP11	*Jumper*	Utilizá-lo fechado quando necessário comunicação RS232 full-duplex no conector DB2
28	P13	*USB*	Conector de saída da comunicação *USB*

B.5 MÓDULO PRINCIPAL DO ARM (ARM-CORE)

A Figura 58 mostra o posicionamento dos principais componentes que compõem o núcleo de processamento associado ao *ARM*.

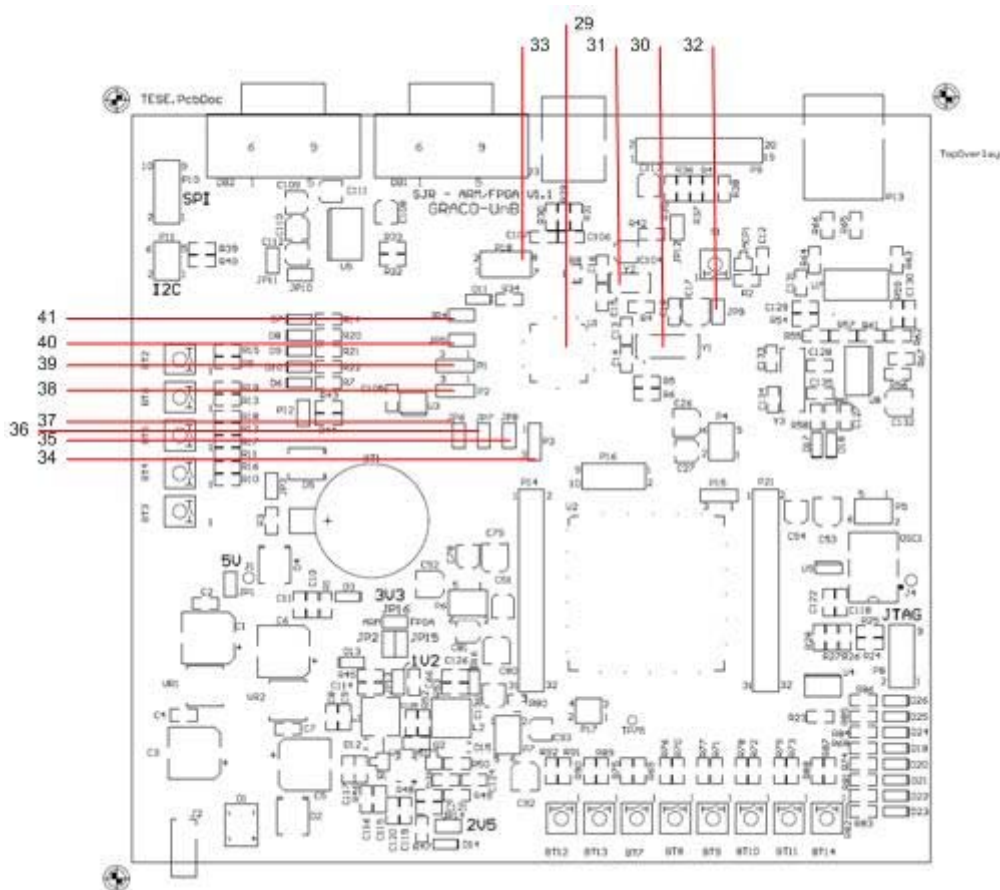



Figura 58 – Localização dos componentes do módulo principal do *ARM – CORE*.

Tabela 25 – Descrição dos componentes referentes ao módulo *ARM – CORE*.

Item	*Nome*	*DESCRIÇÃO*	*FUNÇÃO*
29	U1	*LPC2146FBD64*	Microcontrolador *ARM7*
30	Y1	*Xtal*	Cristal oscilador 14.7456Mhz
31	Y2	*Xtal*	Cristal oscilador 32.768KHz
32	JP9	*Jumper*	Quando desconectado, isola a voltagem de referência, *Vreff*, no microcontrolador. O padrão da placa é de 3,3V advindo do regulador LM7833. Este *jumper* foi inserido para dar a possibilidade de utilizar outros níveis de tensões para referência dos conversores ADC
33	P18	*Conector*	Disponibiliza os pinos não utilizados para outras funções, conforme necessidade do usuário.
34	P3	*Seletor*	Quando os pinos 1 e 2 são conectados, utilizando-se *jumper,* habilita-se a ligação direta do pino *SDA* da comunicação I^2C com o *FPGA*. Quando os pinos 2 e 3 são conectados, utilizando-se *jumper,* habilita-se a seleção do *boot loader select* – da interface *ISP/IAP*.
35	JP8	*Jumper*	Quando conectado, habilita-se a ligação direta do pino SCL da comunicação I^2C com o *FPGA*.
36	JP7	*Jumper*	Quando conectado, habilita-se a ligação direta da função *RX1* do *ARM* com o *FPGA*.
37	JP6	*Jumper*	Quando conectado, habilita-se a ligação direta da função *TX1* do *ARM* com o *FPGA*.
38	P2	*Seletor*	Quando os pinos 1 e 2 são conectados, utilizando-se *jumper,* habilita-se a ligação direta do pino *SDA* com a memória, U3. Quando os pinos 2 e 3 são conectados, utilizando-se *jumper,* habilita-se a ligação direta do pino *SDA* e o conector de borda da comunicação I^2C, P11.
39	P1	*Seletor*	Quando os pinos 1 e 2 são conectados, utilizando-se *jumper,* habilita-se a ligação direta do pino *SCL* com a memória, U3. Quando os pinos 2 e 3 são conectados, utilizando-se *jumpers,* habilita-se a ligação direta do pino *SCL* e o conector de borda da comunicação I^2C, P11.
40	JP5	*Jumper*	Quando conectado, habilita-se a ligação direta da função *RX0* do *ARM* com o *FPGA*.
41	JP4	*Jumper*	Quando conectado, habilita-se a ligação direta da função *TX0* do *ARM* com o *FPGA*.

B.6 ALIMENTAÇÃO DO FPGA (1V2, 2V5, 3V3)

A Figura 59 mostra o posicionamento dos principais componentes que constituem o sistema de alimentação do *FPGA*.

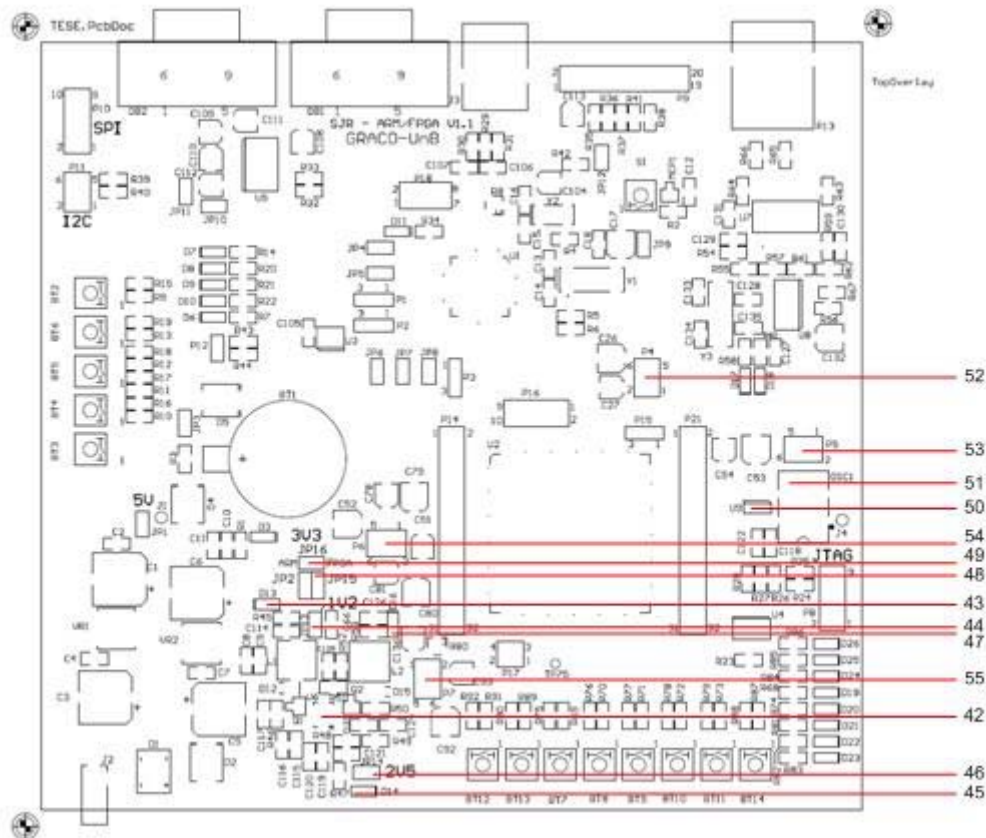



Figura 59 – Localização dos componentes do módulo de alimentação do *FPGA – Alimentação*.

Tabela 26 – Descrição dos componentes referentes ao módulo *FPGA – Alimentação*.

Item	*Nome*	*DESCRIÇÃO*	*FUNÇÃO*
42	U6	*TPS75003*	Regulador de tensões – 1V2, 2V5 e 3V3.
43	D13	*LED*	Indicação visual de que a alimentação 1V2 está ativa.
44	JP13	*Jumper*	Quando desconectado, isola o circuito de suprimento de energia em nível de tensão 1V2.
45	D14	*LED*	Indicação visual de que a alimentação 2V5 está ativa.

46	JP14	*Jumper*	Quando desconectado, isola o circuito de suprimento de energia em nível de tensão 2V5.
47	D16	*LED*	Indicação visual de que a alimentação 3V3 está ativa.
48	JP15	*Jumper*	Quando desconectado, isola o circuito de suprimento de energia em nível de tensão 3V3 do *FPGA.*
49	JP16	*Jumper*	Quando conectado, compartilha a alimentação do regulador LM7833 entre o *ARM* e o *FPGA*. Neste caso, ambos serão alimentados com a tensão 3,3V proveniente do regulador LM7833.
50	U9	*Clock Distributor*	Distribuidor de *clock* entre os *Banks* do *FPGA*
51	OSC1	*Oscilator*	Oscilador de 50MHz
52	P4	*Seletor de voltagem*	Quando os pinos 1 e 2 são conectados, por meio de *jumper*, o *Bank0* é alimentado com a tensão em nível 1V2. Quando os pinos 3 e 4 são conectados, o *Bank0* é alimentado com a tensão em nível 2V5. Quando os pinos 5 e 6 são conectados, o *Bank0* é alimentado com a tensão em nível 3V3. **NOTA: Apenas uma das configurações deve ser utilizada de cada vez. Recomenda-se desligar o *FPGA* para alterar configurações de alimentação.**
53	P5	*Seletor de Voltagem*	Quando os pinos 1 e 2 são conectados, por meio de *jumper,* o *Bank1* é alimentado com a tensão em nível 1V2. Quando os pinos 3 e 4 são conectados, o *Bank1* é alimentado com a tensão em nível 2V5. Quando os pinos 5 e 6 são conectados, o *Bank1* é alimentado com a tensão em nível 3V3. **NOTA: Apenas uma das configurações deve ser utilizada de cada vez. Recomenda-se desligar o *FPGA* para alterar configurações de alimentação.**
54	P6	*Seletor de Voltagem*	Quando os pinos 1 e 2 são conectados, por meio de *jumper,* o *Bank3* é alimentado com a tensão em nível 1V2. Quando os pinos 3 e 4 são conectados, o *Bank3* é alimentado com a tensão em nível 2V5. Quando os pinos 5 e 6 são conectados, o *Bank3* é alimentado com a tensão em nível 3V3. **NOTA: Apenas uma das configurações deve ser utilizada de cada vez. Recomenda-se desligar o *FPGA* para alterar configurações de alimentação.**

55	P7	*Seletor de Voltagem*	Quando os pinos 1 e 2 são conectados, por meio de *jumper,* o *Bank2* é alimentado com a tensão em nível 1V2. Quando os pinos 3 e 4 são conectados, o *Bank2* é alimentado com a tensão em nível 2V5. Quando os pinos 5 e 6 são conectados, o *Bank2* é alimentado com a tensão em nível 3V3. **NOTA: Apenas uma das configurações deve ser utilizada de cada vez. Recomenda-se desligar o *FPGA* para alterar configurações de alimentação.**

B.7 MÓDULO DE MEMÓRIA PARA O FPGA (FPGA - MEMÓRIA)

A Figura 60 mostra o posicionamento da memória associada ao *FPGA*.

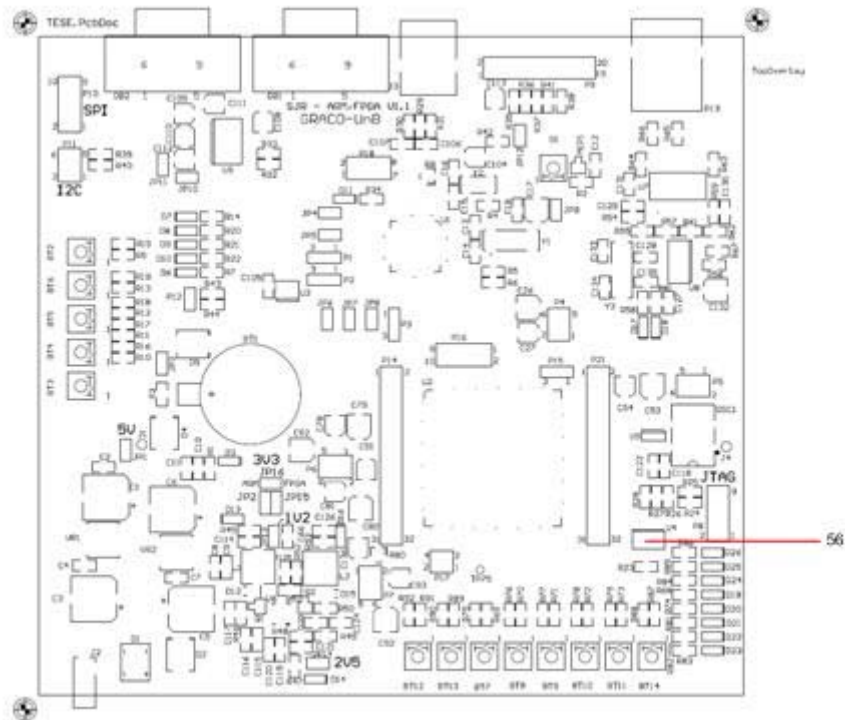



Figura 60 – Localização dos componentes da memória do módulo *FPGA – Memória***.**

Tabela 27 – Descrição dos componentes referentes ao módulo *FPGA – Memória*.

Item	*Nome*	*DESCRIÇÃO*	*FUNÇÃO*
56	U4	*Circuito Integrado –*	Memória externa do *FPGA*. Capacidade 4M*byte*.

		XCF04SVOG20C	

B.8 PROJETO DE INTERFACES PARA O FPGA

A Figura 61 mostra o posicionamento dos dispositivos de interfaces (botões, *LEDs* e potenciômetro) ligados ao *FPGA*.

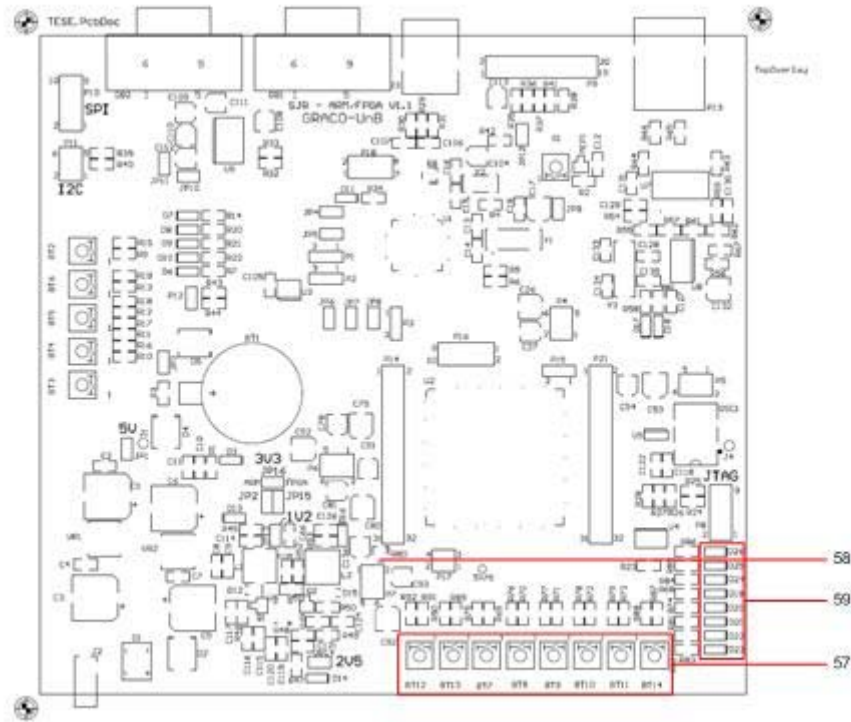



Figura 61 – Localização dos componentes do módulo *FPGA – INTERFACES*.

Tabela 28 – Descrição dos componentes referentes ao módulo *FPGA – INTERFACES*.

Item	***Nome***	***DESCRIÇÃO***	***FUNÇÃO***
57	BT7-BT4	*Botões*	Interface de botões com o usuário
58	R80	*Potenciômetro*	Potenciômetro de 10kΩ
59	LED19-LED26	*LEDs*	Interface de *LEDs* com o usuário

B.9 PROJETO DE INTERFACES DO FPGA (FPGA –JTAG/ ETHERNET)

A Figura 62 mostra o posicionamento dos dispositivos que compõem a interface *JTAG* e *Ethernet* ligados ao *FPGA*.

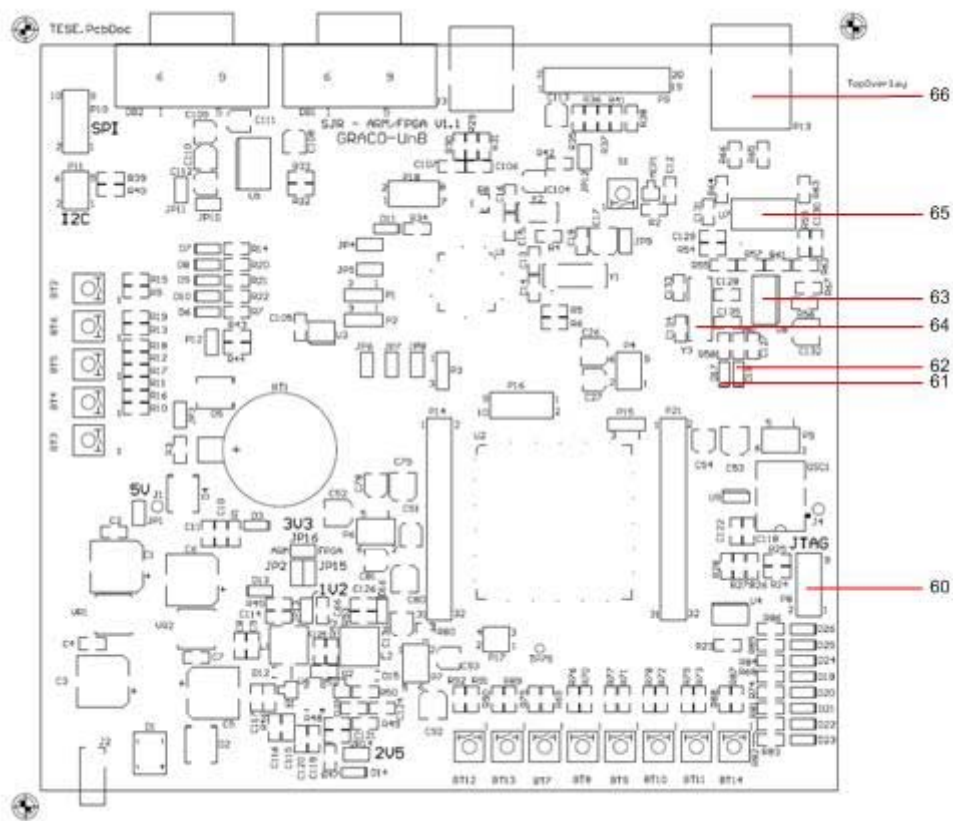



Figura 62 – Localização dos componentes do módulo *FPGA – JTAG/ETHERNET*.

Tabela 29 – Descrição dos componentes referentes ao módulo *FPGA – JTAG/ETHERNET*.

Item	***Nome***	***DESCRIÇÃO***	***FUNÇÃO***
60	P8	*Conector JTAG*	Disponibiliza os pinos para comunicação *JTAG*.
61	D17	*LED*	Sinalizador visual de utilização da interface ETHERNET.
62	D18	*LED*	Sinalizador visual de utilização da interface ETHERNET.
63	U8	*Circuito Integrado ENC28J60*	Driver da comunicação ETHERNET.
64	Y3	*XTal*	Cristal oscilador 25MHz.
65	U7	*Pulse Transformer*	Transfomador de pulso. Necessário à comunicação ETHERNET.

66	P13	*USB*	*Conector Ethernet*.

B.10 MÓDULO PRINCIPAL DO FPGA SPARTAN 3 (FPGA - CORE)

A Figura 63 mostra o posicionamento dos principais componentes que compõem o núcleo de processamento associado ao *FPGA*.

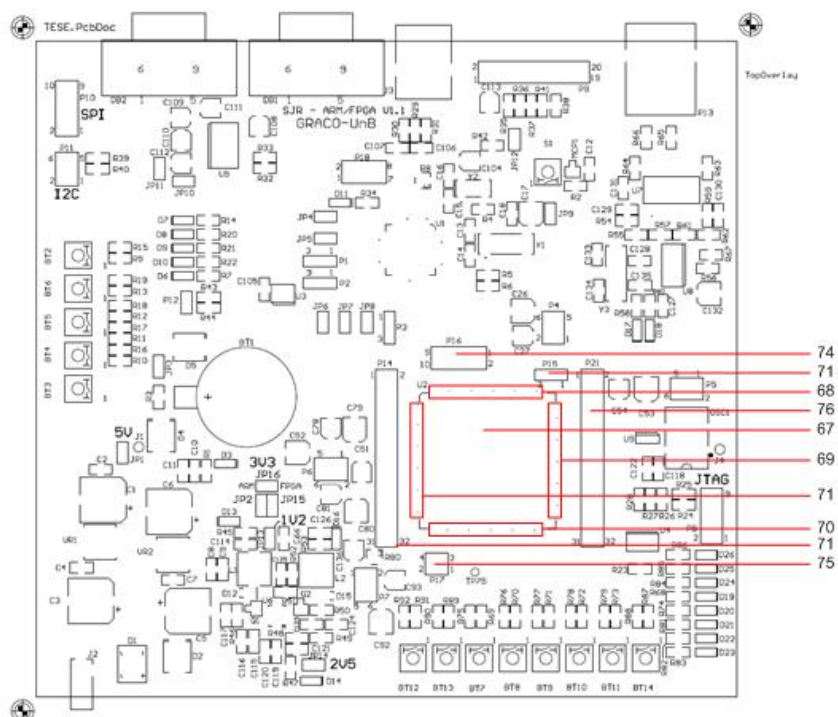



Figura 63 – Localização dos componentes do módulo *FPGA – CORE*.

Tabela 30 – Descrição dos componentes referentes ao módulo *FPGA – CORE*.

Item	***Nome***	***DESCRIÇÃO***	***FUNÇÃO***
67	U2	*Circuito Integrado XC3S500E-4PQG208C*	Unidade de processamento *FPGA*
68	-	*Bank0*	Disposição do *Bank0* na *FPGA*
69	-	*Bank1*	Disposição do *Bank1* na *FPGA*
70	-	*Bank2*	Disposição do *Bank2* na *FPGA*

71	-	*Bank3*	Disposição do *Bank3* na *FPGA*
72	P14	*Conector*	Disponibiliza pinos não utilizados do *Bank3*
73	P15	*Conector*	Disponibiliza pinos não utilizados do *Bank0*
74	P16	*Conector*	Disponibiliza pinos não utilizados do *Bank0*
75	P17	*Conector*	Disponibiliza pinos não utilizados do *Bank2*
76	P21	*Conector*	Disponibiliza pinos não utilizados do *Bank1*

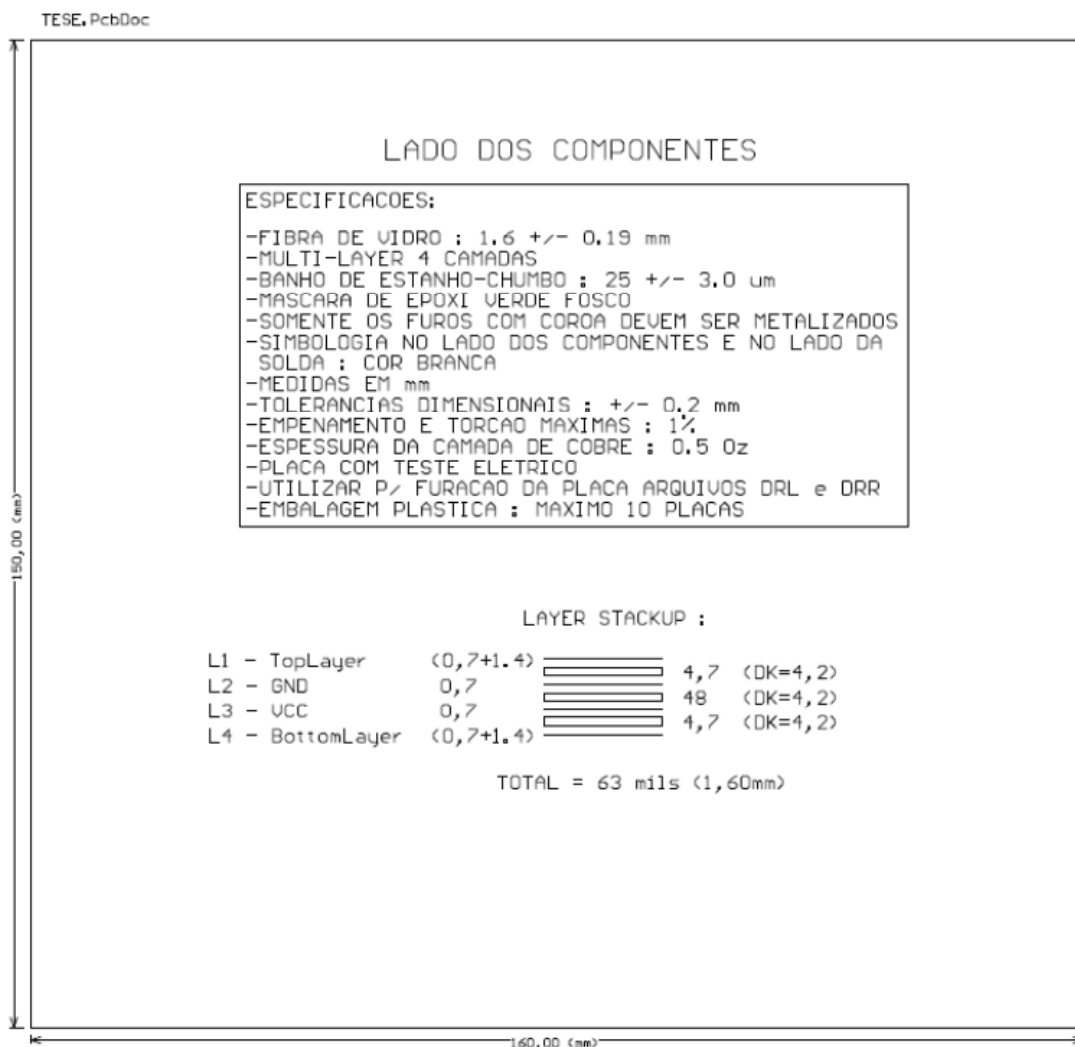
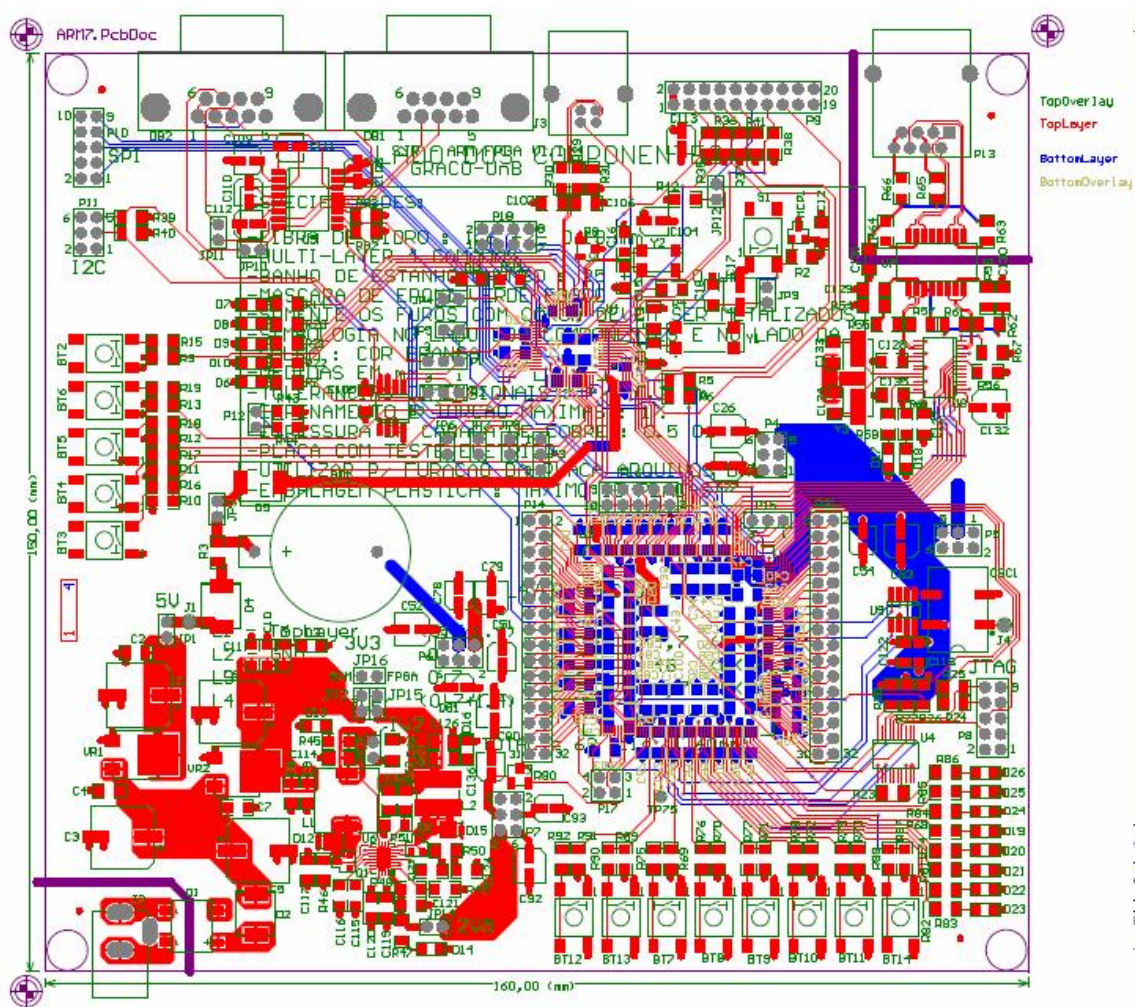
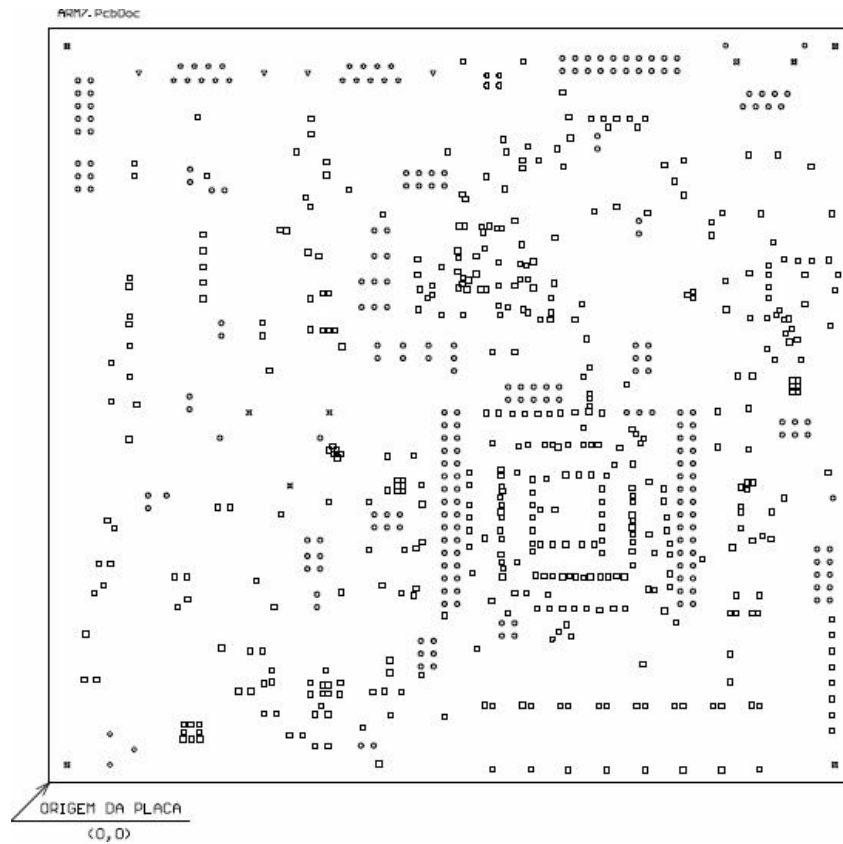
B.11 CONCLUSÃO SOBRE A APRESENTAÇÃO DA PLATAFORMA

Esta sessão foi dedicada a apresentar fisicamente os componentes dos módulos implementados na plataforma e as funções dos conectores e principais dispositivos. O objetivo aqui foi apresentar os módulos e seus componentes para que o desenvolvedor ou usuário possa localizar mais facilmente os componentes constituintes do sistema.

Assim, este Anexo servirá de guia e documentação de suporte necessária quando da utilização da plataforma. Isso porque além de servir como mapa referenciado do sistema ainda traz um resumo de todas as funções dos dispositivos mais relevantes e *jumper* a serem constantemente utilizados.

APÊNDICE C. APRESENTAÇÃO DOS ARQUIVOS DE LAYOUT DA PLATAFORMA - PCI

A seguir são apresentados todos os *layouts* elaborados pra a confecção da plataforma. A placa foi fabricada em quatro camadas, sendo: L1 a camada com trilhas na parte superior, L2 com todo o plano de terra, L3 a camada de alimentação e, por último, L4 com trilhas da parte inferior da placa.

C.1 PCI – RECOMENDAÇÕES GERAIS

C.2 PCI – LAYOUT COMPLETO

C.3 PCI – PLANO DE FUROS

Symbol	Hit Count	Tool Size	Physical Length	Rout Path Length	Plated	Hole Type
□	467	16mil (0.406mm)			PTH	Round
▷	5	35mil (0.889mm)			PTH	Round
○	214	40mil (1.016mm)			PTH	Round
⊙	18	47mil (1.194mm)			PTH	Round
⊙	2	63mil (1.6mm)			PTH	Round
⌗	3	71mil (1.803mm)			NPTH	Round
□	2	100mil (2.54mm)			NPTH	Round
⊞	6	126mil (3.2mm)			NPTH	Round
▽	4	140mil (3.556mm)			PTH	Round
◇	3	50mil (1.27mm)	120mil (3.048mm)	70mil (1.778mm)	PTH	Slot
	724 Total					

Slot definitions : Rout Path Length = Calculated from tool start centre position to tool end centre position.
Physical Length = Rout Path Length + Tool Size = Slot length as defined in the PCB layout

DrillDrawing

C.4 PCI – Camada 1 – TOP LAYER

C.5 PCI – Camada 2 – GND

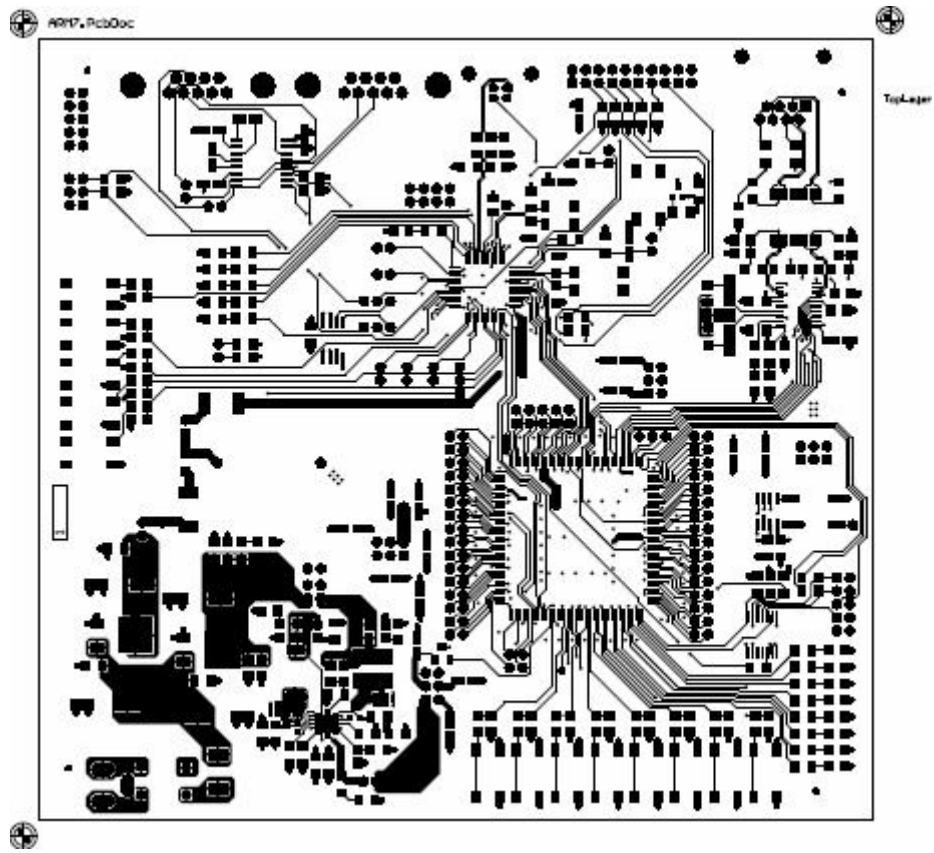
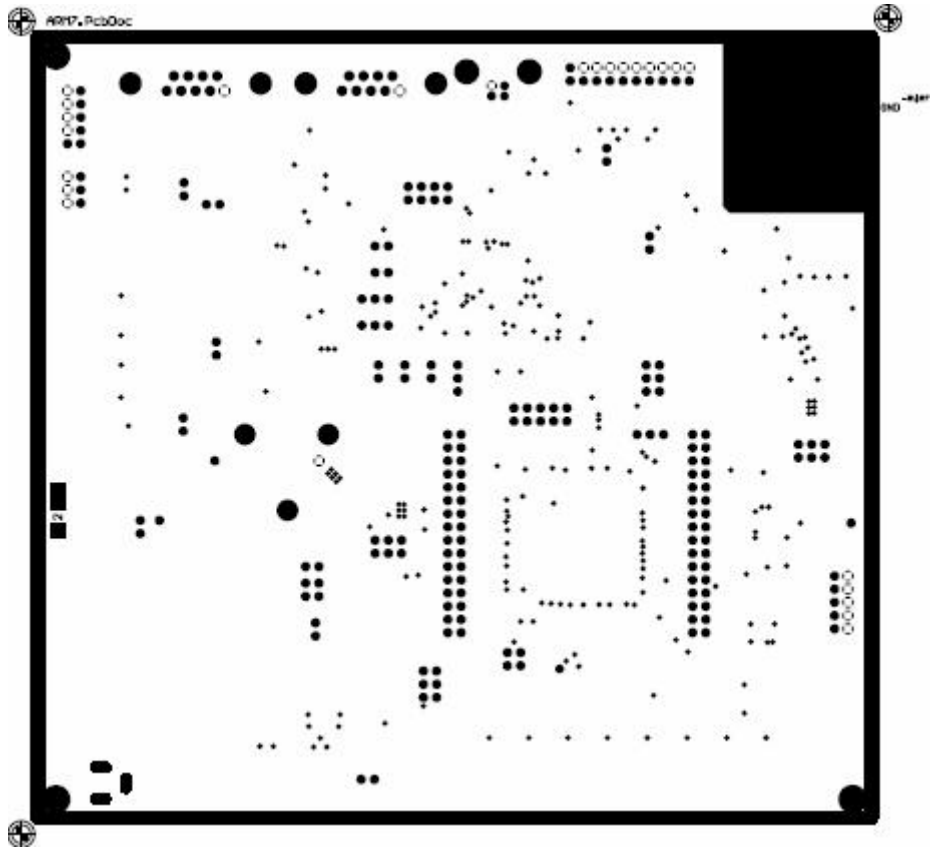
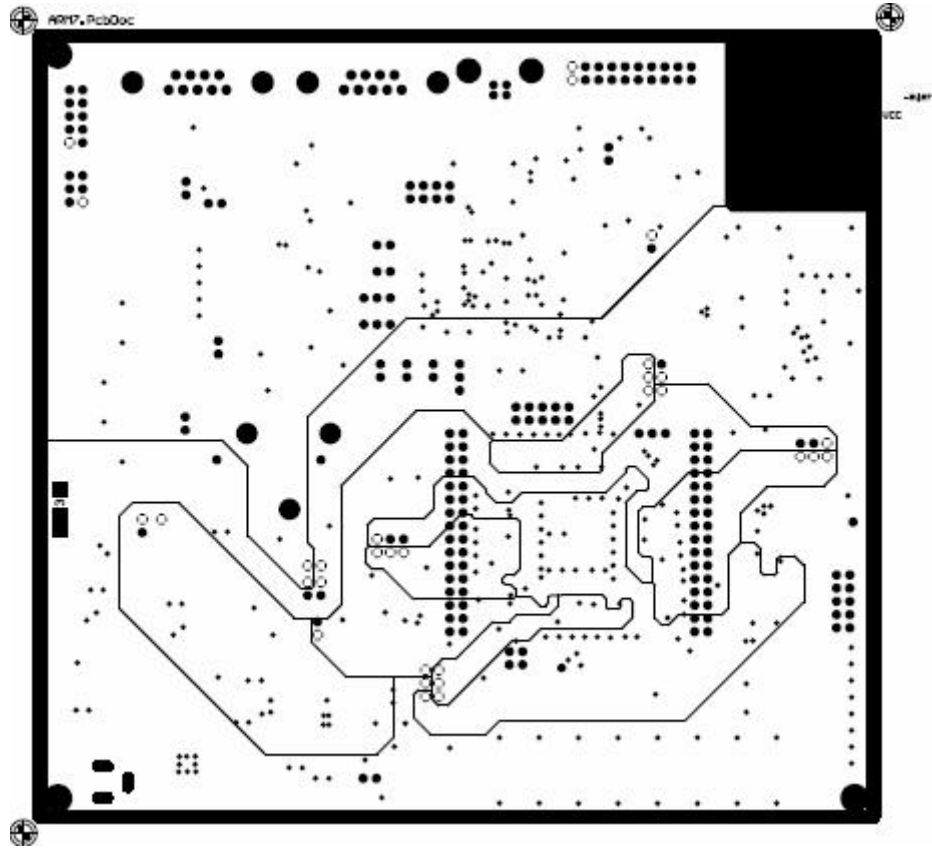
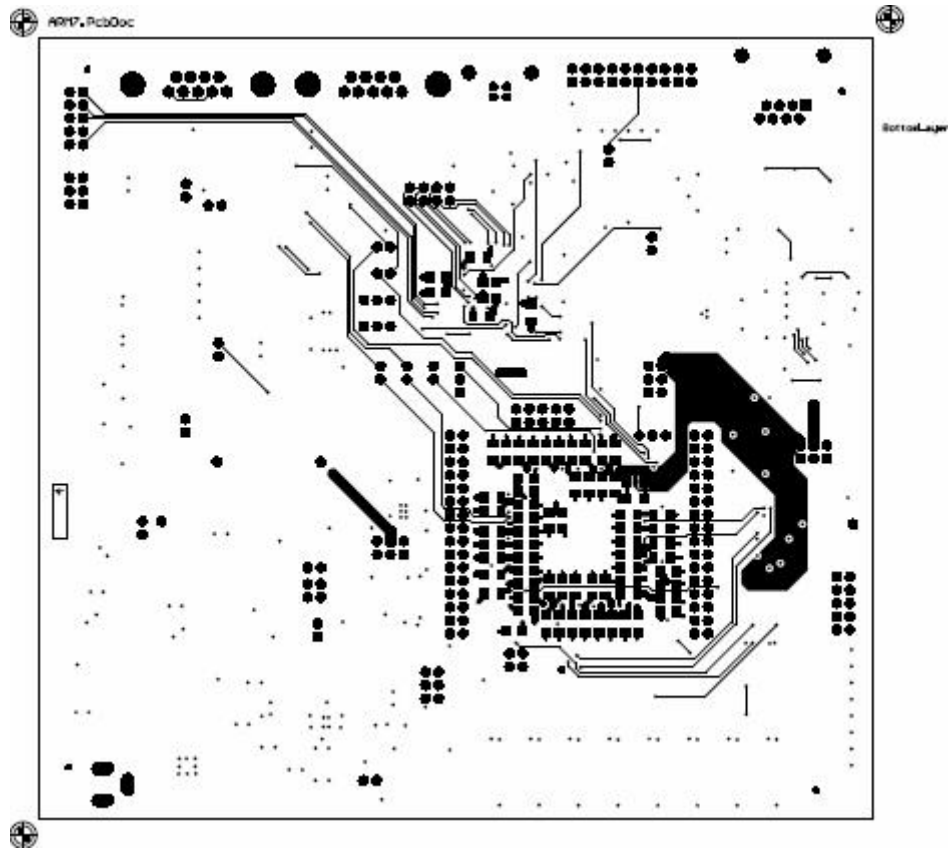
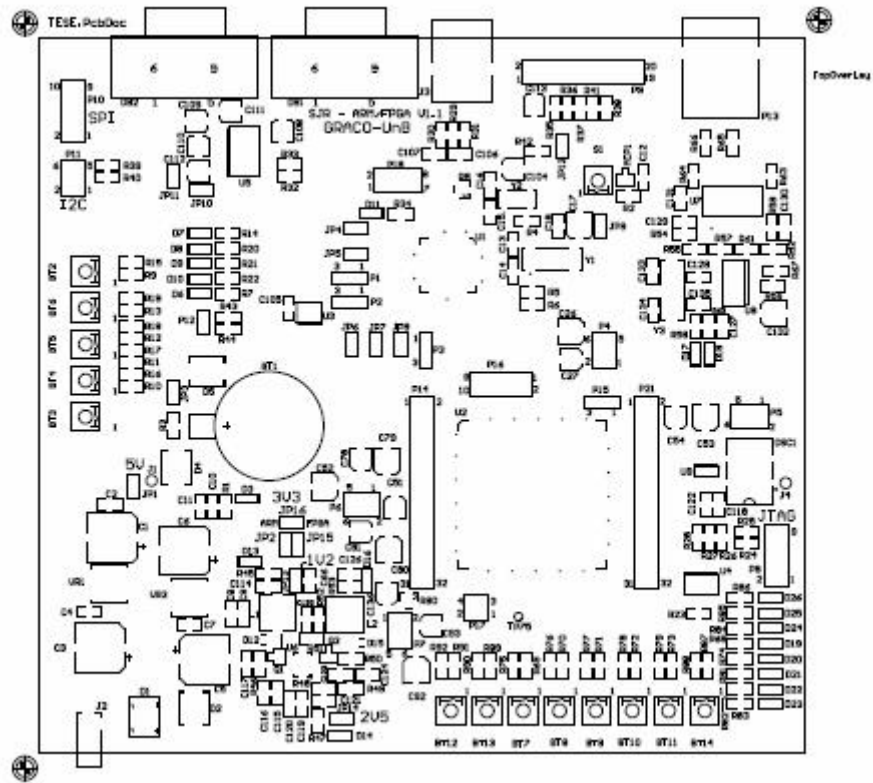
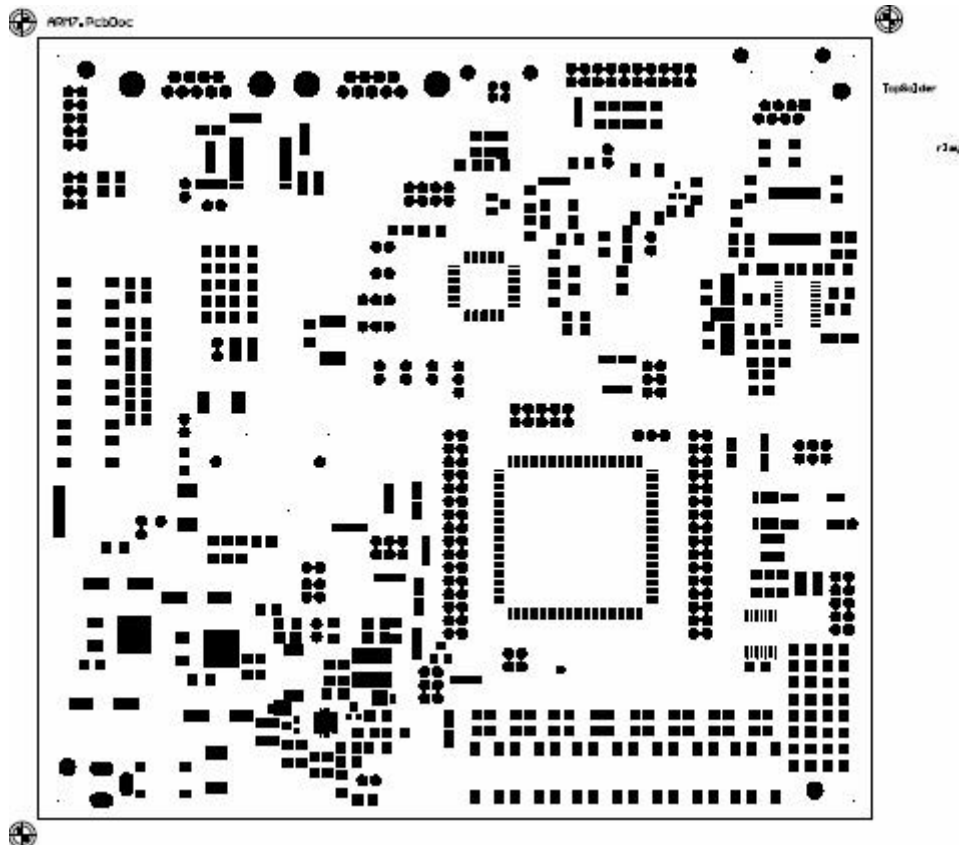
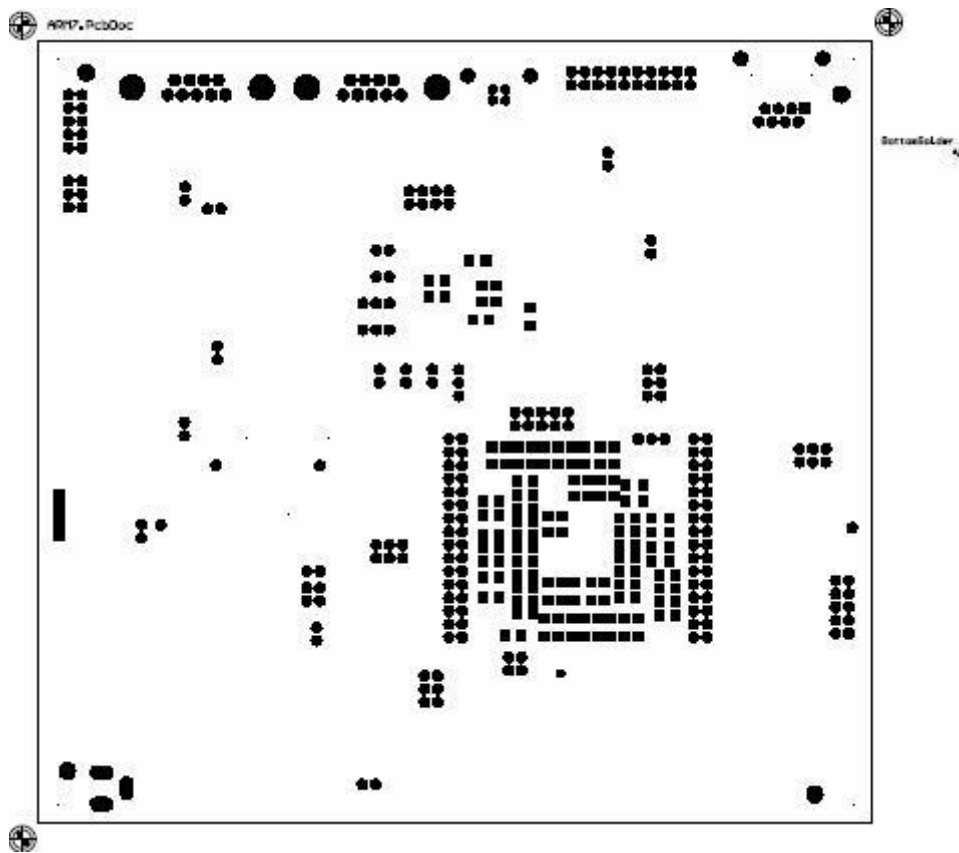
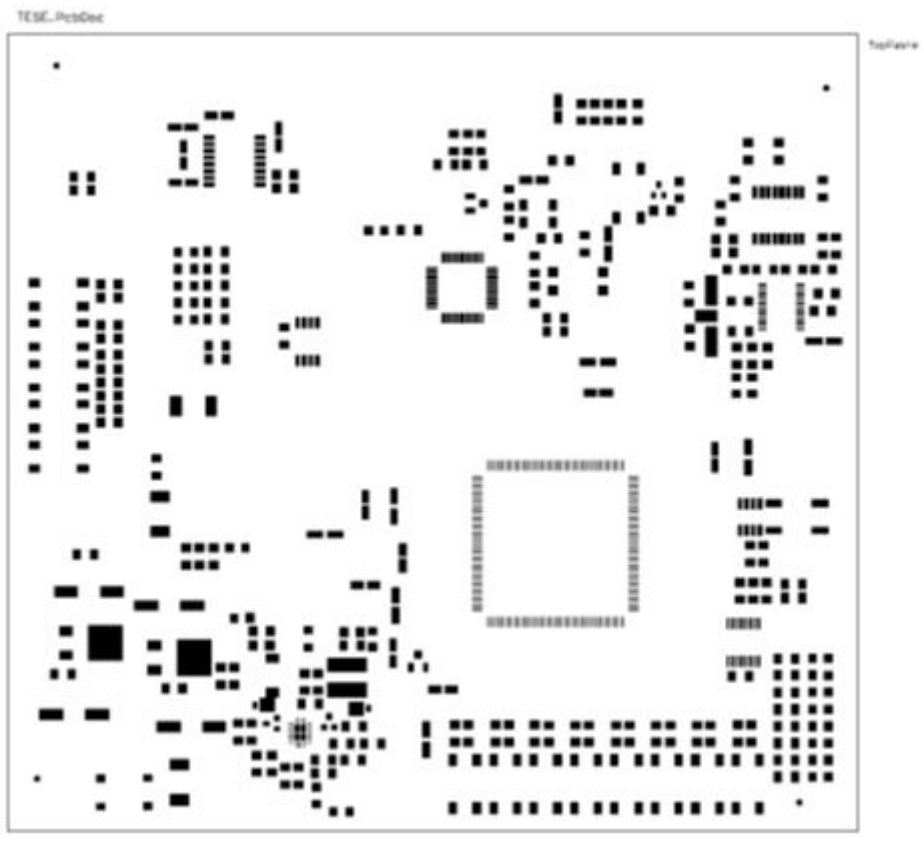
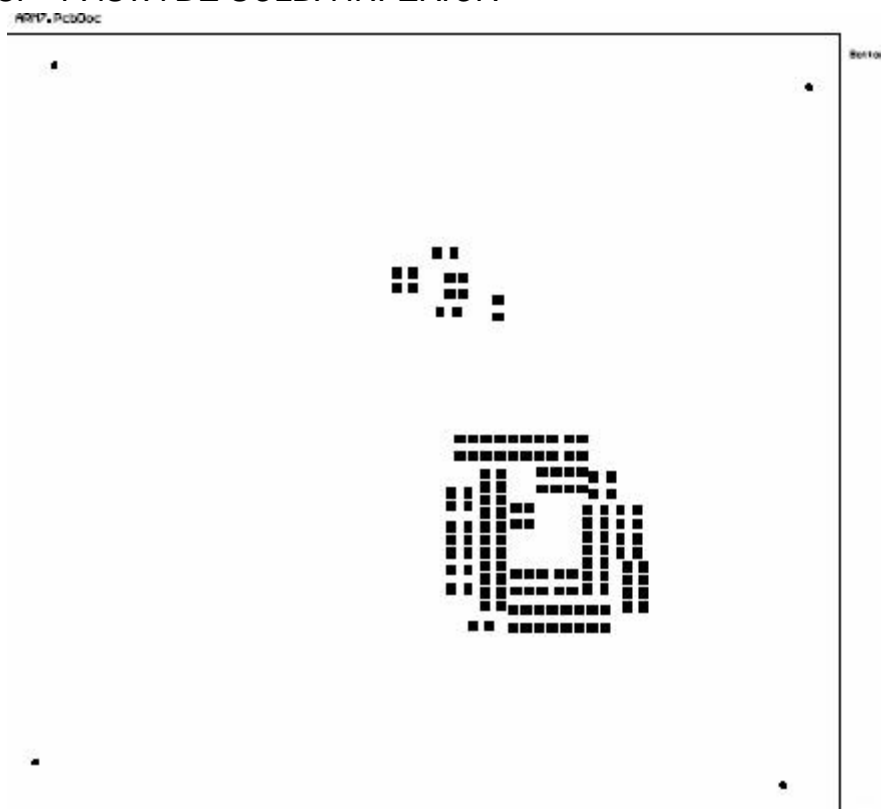
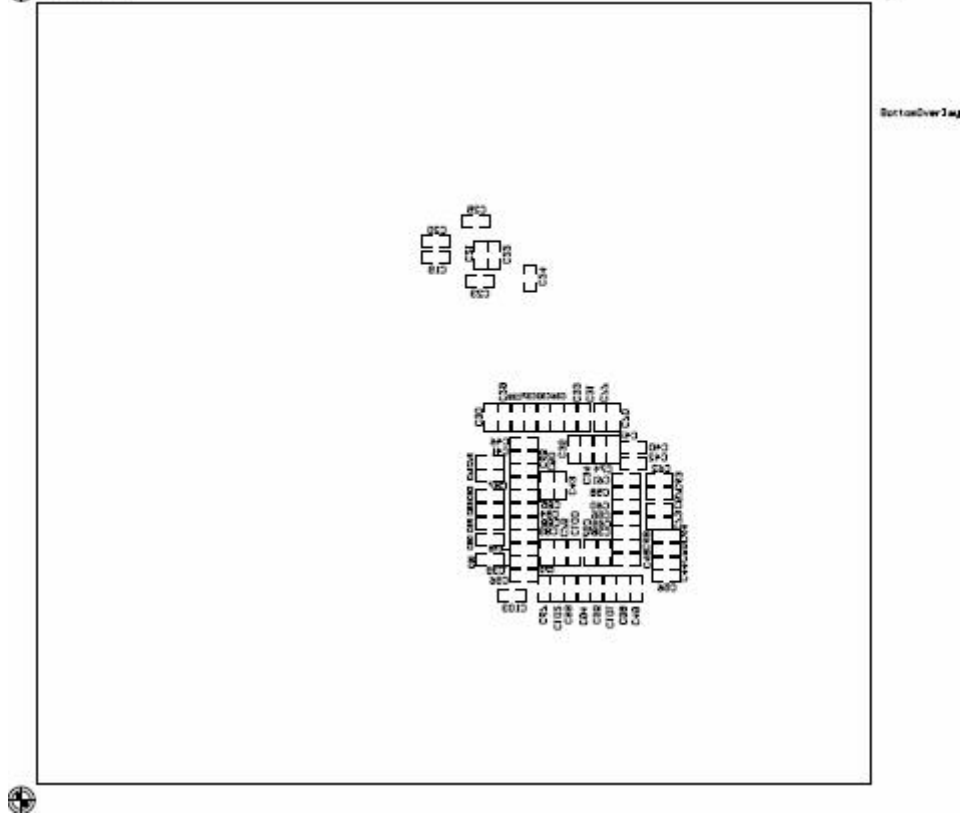
ARM7.PcbDoc

C.6 PCI – Camada 3 – VCC

C.7 PCI – Camada 4 – BOTTON LAYER

C.8 PCI – MÁSCARA SUPERIOR - SILK

C.9 PCI – MÁSCARA DE SOLDA SUPERIOR

C.10 PCI – MÁSCARA DE SOLDA INFERIOR

C.11 PCI – PASTA DE SOLDA SUPERIOR

C.12 PCI – PASTA DE SOLDA INFERIOR

C.13 PCI – MÁSCARA INFERIOR - SILK

APÊNDICE D. LISTA DOS COMPONENTES DA PLATAFORMA

Tabela C.1 – Lista de componentes para a fabricação de uma unidade da plataforma reconfigurável.

Qde	*Part Number*	*Comentário*	*Valor*	*Descrição*	*Designador*
1	BS-3	Battery Holder	3.6V	Multicell Battery	BT1
11	FSM2JSMA	Push Button	Push Button	Switch	BT2, BT3, BT4, BT5, BT6, BT7, BT8, BT9, BT10, BT11, S1
7	AVE477M16G24T-F	470uF/16V	470uF/16V	Polarized Capacitor (Radial)	C1, C3, C5, C6, C109, C110, C112
22	12065C104KAT2A	0.1uF	0.1uF	Capacitor (Semiconductor SIM Model)	C2, C4, C7, C8, C9, C10, C11, C12, C18, C20, C22, C24, C25, C105, C117, C118, C122, C124, C128, C129, C130, C135
2	CC1206JRNP09BN390	39pF	39pF	Capacitor (Semiconductor SIM Model)	C13, C14
2	CC1206DRNP09BN6R8	6.8pF	6.8pF	Capacitor (Semiconductor SIM Model)	C15, C16
8	AVE106M35C12T-F	10uF / 35V	10uF / 35V	Polarized Capacitor (Radial)	C17, C26, C52, C53, C79, C80, C92, C132
3	CC1206JRNP09BN101	100pF	100pF	Capacitor (Semiconductor SIM Model)	C19, C21, C23
6	AVE474M50B12T-F	0.47uF	0.47uF	Polarized Capacitor (Radial)	C27, C51, C54, C78, C81, C93

30	CC1206KRX7R9BB102	1nF	1nF	Polarized Capacitor (Radial)	C28, C29, C30, C31, C32, C46, C47, C48, C49, C50, C55, C56, C57, C58, C59, C73, C74, C75, C76, C77, C82, C83, C84, C85, C86, C94, C95, C96, C97, C98
37	CC1206KRX7R9BB473	47nF	47nF	Polarized Capacitor (Radial)	C33, C34, C35, C36, C37, C38, C39, C40, C41, C42, C43, C44, C45, C60, C61, C62, C63, C64, C65, C66, C67, C68, C69, C70, C71, C72, C87, C88, C89, C90, C91, C99, C100, C101, C102, C103, C127
5	AVE104M50B12T-F	0.1uF/50V	0.1uF/50V	Polarized Capacitor (Radial)	C104, C108, C111, C113, C136
4	CC1206JRNP09BN180	18pF	18pF	Capacitor (Semiconductor SIM Model)	C106, C107, C133, C134
2	TLJG107M006R0800	Tantalum 100uF	100uF	Capacitor (Semiconductor SIM Model)	C114, C126
1	CC1206KRX7R9BB103	0.01uF	0.01uF	Capacitor (Semiconductor SIM Model)	C115
2	CC1206KRX7R9BB152	1500pF	1500pF	Capacitor (Semiconductor SIM Model)	C116, C123

1	GRM31CF50J107ZE01L	100uF	100uF	Capacitor (Semiconductor SIM Model)	C119
1	C3216X7R1C105M/0.85	1uF	1uF	Capacitor (Semiconductor SIM Model)	C120
1	CC1206ZKY5V7BB106	10uF	10uF	Capacitor (Semiconductor SIM Model)	C121
1	CC1206JRNP09BN100	10pF	10pF	Capacitor (Semiconductor SIM Model)	C125
1	202R18W102KV4E	1nF 2kV	1nF 2kV	Capacitor (Semiconductor SIM Model)	C131
1	ICS551MLFT	CLK_Dist	CLK_Dist		CLK_Dist1
1	SDB104-TP	Bridge	Bridge	Full Wave Diode Bridge	D1
3	SMLJ60S6-TP	Diode	Diode	Default Diode	D2, D4, D5
17	APL3015QBC/D-F01	LED	LED	Typical INFRARED GaAs LED	D3, D6, D7, D8, D9, D10, D11, D13, D14, D16, D17, D18, D19, D20, D21, D22, D23
1	SS2P2L-E3/84A	Diode SS2P2L	SS2P2L	Silicon AF Schottky Diode for High-Speed Switching	D12
1	MBRM120LT3G	Diode MBRM120	MBRM120	Silicon AF Schottky Diode for High-Speed Switching	D15
2	1734348-1	DB9-F	DB9-F	Receptacle Assembly, 9 Position, Right Angle	DB1, DB2
2	Pino unico para testes 180º	Socket		Socket	J1, J4
1	PJ-031D	Jack - PWR	Up to 12V	Low Voltage Power Supply Connector	J2
1	690-004-221-023	Jack *USB*	*USB* 2.0	*USB* 1.1, Right Angle, Thru-Hole, B Type, Receptacle, 4 Position, Black	J3

23	Barra de Pinos	1V2, 2V5, 3V3, 5V, 5X2, BSD, CTS, I2C *ARM*, *JTAG*, *JTAG ARM*, RTS, RX0, RX1, SCL_I2C, SDA_I2C, SLC, SPI *ARM*, TX0, TX1, Vbat, Vref	1X2, 1X3, 2X5, 2X6, 2X10	Header, 2-Pin, Header, 3-Pin, Header, 3-Pin, Dual row, Header, 5-Pin, Dual row, Header, 10-Pin, Dual row, Jumper Wire	JP1, JP2, JP3, JP4, JP5, JP6, JP7, JP8, JP9, JP10, JP11, JP12, JP13, JP14, JP15, P1, P2, P3, P8, P9, P10, P11, P12
1	CDRH74NP-150MC	15uH	15uH	Inductor	L1
1	SRR6028-5R0Y	5uH	5uH	Inductor	L2
1	MCP130T-315I/TT	MCP130-315	MCP130-315		MCP1
1	ECS-8FA3X-500-TR	50Mhz	50Mhz		OSC1
4	Barra de pinos em T	Header 4		Header, 4-Pin	P4, P5, P6, P7
1	A-2004-2-4-LPS-N-R	RJ45	RJ45	Header, 8-Pin	P13
2	SI2323DS-T1-E3	Mosfet	SI2323	P-Channel MOSFET	Q1, Q2
18	RMCF1206JT270R	270R	270R	Semiconductor Resistor	R1, R7, R14, R20, R21, R22, R28, R34, R45, R47, R53, R58, R60, R68, R74, R81, R82, R83
18	MCR18EZPJ103	10K	10K	Semiconductor Resistor	R2, R9, R10, R11, R12, R13, R35, R36, R37, R38, R41, R42, R43, R69, R70, R71, R72, R73
14	MCR18EZPJ102	1K	1K	Semiconductor Resistor	R3, R15, R16, R17, R18, R19, R32, R33, R44, R75, R76, R77, R78, R79
6	RMCF1206JT68R0	68R	68R	Semiconductor Resistor	R4, R5, R6, R23, R24, R25

2	SM-3TW103	POT 10K	10K	Tapped Resistor	R8, R80
2	RMCF1206JT4K70	4K7	4K7	Resistor	R26, R27
2	RMCF1206JT33R0	33R	33R	Semiconductor Resistor	R29, R30
1	RMCF1206JT1K50	1K5	1K5	Semiconductor Resistor	R31
2	RMCF1206JT2K00	2K	2K	Semiconductor Resistor	R39, R40
2	LRC-LR1206LF-01-R033-F	0.033R	0.033R	Resistor	R46, R50
2	RMCF1206FT61K9	61K9	61K9	Resistor	R48, R52
1	RMCF1206FT15K4	15K4	15K4	Resistor	R49
1	ERJ-8ENF3652V	36K5	36K5	Resistor	R51
1	LF1206A302R-10	Ferrite Bead	Ferrite Bead	Resistor	R54
4	RMCF1206FT49R9	49R9	49R9	Semiconductor Resistor	R55, R57, R61, R62
1	RMCF1206JT100K	100K	100K	Semiconductor Resistor	R56
1	Jumper 3216(1206)	Jumper	0R	Resistor	R59
4	RMCF1206JT75R0	75R	75R	Semiconductor Resistor	R63, R64, R65, R66
1	RMCF1206FT2K32	2K32	2K32	Semiconductor Resistor	R67
1	LPC2146FBD64,151	*ARM7*	LPC2146	*ARM7* 16/32-Bit Microcontroller, 256KB Flash, 32KB RAM (+8KB *USB* DMA shared), 2KB *USB* RAM, 64-Lead LQFP	U1
1	XC3S500E-4PQG208C	XC3S500E-4PQG208C	XC3S500E	Spartan-3E 1.2V *FPGA*, 158 User I/Os, 208-Pin Pb-Free PQFP, Standard Performance, Commercial Grade	U2
1	24LC1025-I/SM	1Mb	1Mb		U3
1	XCF04SVOG20C	Flash PROM 4Mb	4Mb		U4

1	MAX3232CDWR	MAX3232	MAX232	3.0V TO 5.5V, Low-Power, up to 1Mbps, True RS-232 Transceiver Using Four 0.1µF External Capacitor	U5
1	TPS75003RHLR	Volt Reg TPS75003	TPS75003		U6
1	H1102NL	Pulse Transf	Pulse Transf		U7
1	ENC28J60/SS	ENC28J60	ENC28J60		U8
1	AP1084D50L-13	LM7805	LM7805	Voltage Regulator	VR1
1	AP1084D33L-13	LM7833	3V3	Voltage Regulator	VR2
1	ECS-147.4-20-7SX-TR	14.7456Mhz	14.7456Mhz	Crystal Oscillator, 3rd In-Line Lead Version	Y1
1	ECS-.327-6-17X-TR	32.768KHz	32.768KHz	Crystal Oscillator, 3rd In-Line Lead Version	Y2
1	ABLSG-25.000MHZ-D-2-Y-F-T	25MHz	25MHz	Crystal Oscillator, 3rd In-Line Lead Version	Y3

APÊNDICE E. DESCRIÇÃO DE COMO CONFIGURAR AS FERRAMENTAS DE PROGRAMAÇÃO UTILIZADAS NA PLATAFORMA RECONFIGURÁVEL

Pelo fato de a plataforma agregar duas arquiteturas distintas de processamento isso faz com que sejam necessários dois procedimentos independentes de programação e gravação. A seguir são apresentados os procedimentos necessários para a programação do *ARM7* e do *FPGA*.

Vale destacar que os procedimentos descritos e ferramentas utilizadas não são exclusivas e nem mesmo únicas. Outras metodologias compatíveis com as arquiteturas *ARM7* e SPARTAN-3E podem ser utilizadas.

E.1 PROCEDIMENTO PARA PROGRAMAÇÃO DO ARM7

O fabricante do *ARM7* utilizado, NXP, possui um conjunto de ferramentas para facilitar a programação de seus dispositivos. Essas ferramentas foram utilizadas e podem ser encontradas no *site* do fabricante[32].

E.1.1 UTILIZANDO O IAR-MAKEAPP

Inicialmente, utilizou-se o programa *IAR-MakeApp* (Versão: 4.01B) para criar as bibliotecas do dispositivo que serão utilizadas durante a elaboração do projeto.

Ao abrir o programa, ir à guia *Component-library* e arrastar o dispositivo *LPC2148* para a área de trabalho do programa. Ao fazer isso, será aberta uma janela como mostrado na Figura 64. Para tanto, será necessário preencher o campo de frequência de *clock* (*clock-frequency*) com o valor de 14.7256. Utilizar *ponto* e não *vírgula*. No campo *User-defined-name* basta preencher com o nome do arquivo. Clicar em OK.

[32] www.nxp.com

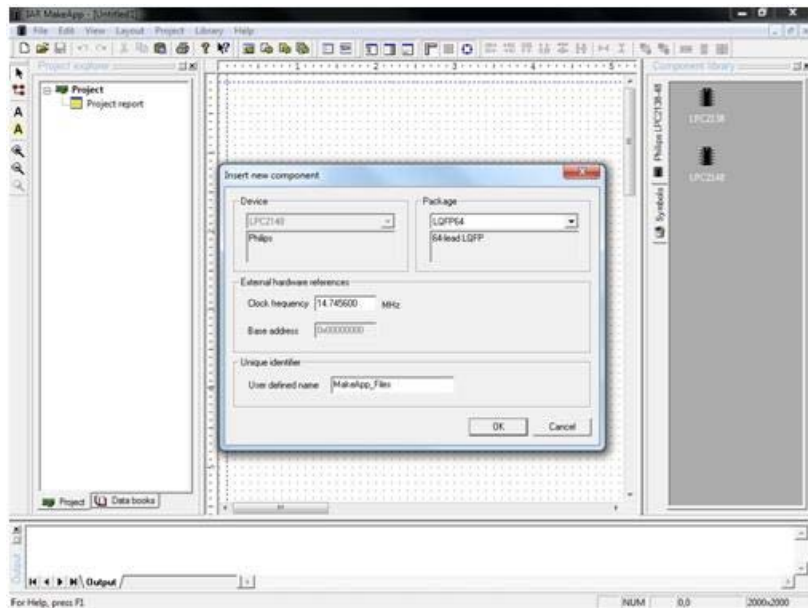



Figura 64 – Configuração inicial - *IAR MakeApp*.

Em seguida, o componente é apresentado na área de trabalho do programa. Pode ser observado que no lado esquerdo do programa já são apresentadas todas as bibliotecas que a serem criadas. Vide Figura 65.

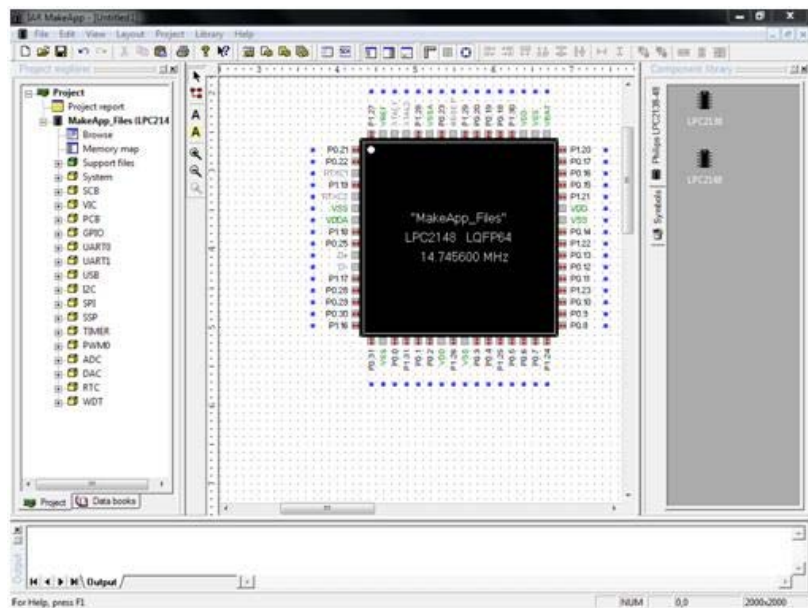



Figura 65 – Apresentação das bibliotecas – *IAR MakeApp***.**

Em seguida, deve-se clicar em *Project >> Settings,* e será aberta uma nova janela, a qual deverá ser preenchida – com (a) o nome do projeto ao qual as bibliotecas serão vinculadas (b) o diretório onde serão salvas e (c) o nome do autor com alguma descrição que comporá a documentação delas. A seguir clicar em OK.

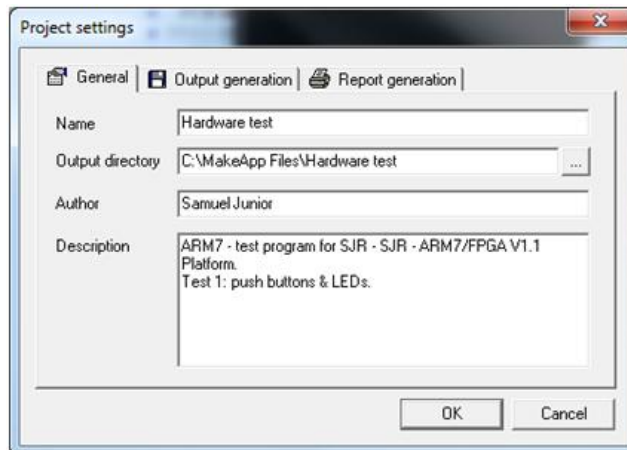



Figura 66 – Suporte à documentação – *IAR MakeApp***.**

Em seguida deve-se expandir a biblioteca de *entradas* e *saídas* de *propósito geral* - GPIO[33], dar um duplo clique em *settings* e configurar os *ports* como entrada ou saída conforme a necessidade.

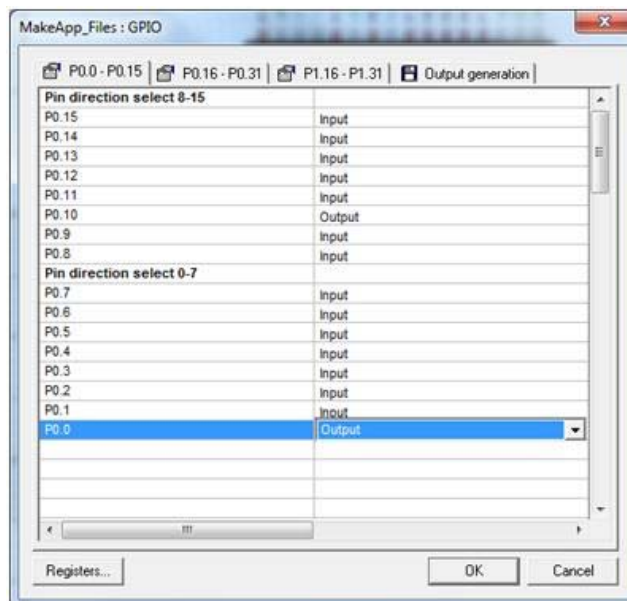



Figura 67 – Configuração de entradas e saídas – IAR MakeApp**.**

Clicar na aba *Output-generation,* dar um clique simples em *ma_gpio.c* e em seguida clicar em *Generate-all-functions. Clicar OK* para concluir o procedimento e fechar a janela.

[33] Em inglês, General Purpose Input Output.

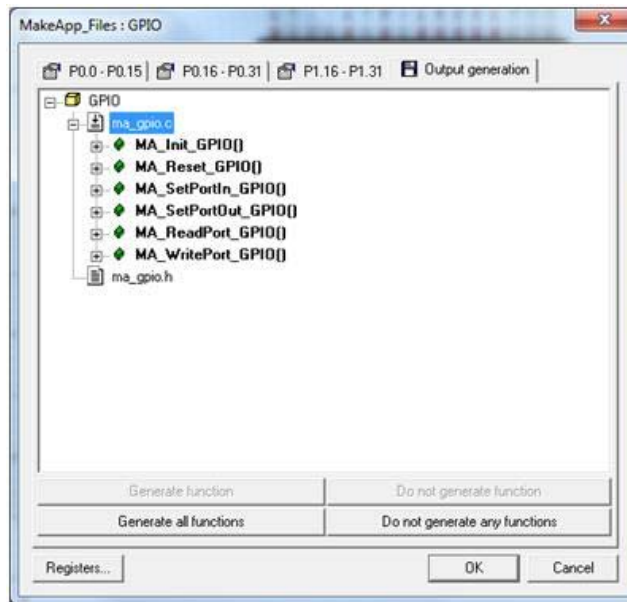



Figura 68 – Gerar todas as funções – *IAR MakeApp*.

Por último, voltar à guia *Project >> Generate-all.* Ao executar esse comando o *IAR-MakeApp* irá gerar todas as bibliotecas apresentadas na pasta de destino, bem como o relatório do projeto. Deve ser observado se a última linha da janela *output view* (parte inferior da área de trabalho) deve apresentar a seguinte descrição: *0 errors and 0 warnings,* conforme pode ser visto na Figura 69*.*

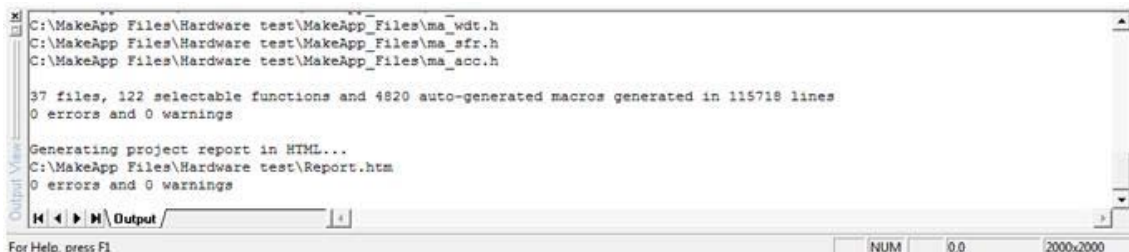



Figura 69 – Relatório de procedimento – *IAR MakeApp*.

E.1.2 UTILIZANDO O C IAR EMBEDDED-WORKBENCH

Feita a configuração no *IAR MakeApp* parte-se então para a configuração do compilador *C IAR Embedded Workbench – IAR EWB.* Antes de se utilizar o compilador *IAR EWB* é necessário fazer algumas alterações para que o ele consiga compilar o programa com os arquivos gerados a partir do *IAR MakeApp*.

De início, é necessário ir até à pasta onde foram salvos os arquivos pelo *MakeApp* e renomear os arquivos *template.c* para *usercode.c* e o arquivo

template.h para *usercode.h*. É necessário ainda adicionar o arquivo *KickStartCard_cstartup.s79* à pasta em que foram geradas as funções usando o *IAR MakeApp,* o qual se encontra no arquivo *ARM7*.exe disponível no site da Editora Érica[34] ou com o autor deste trabalho[35]. Este arquivo tem um código de inicialização do *ARM7* necessário para rodar o programa.

A seguir, parte-se para a configuração do compilador propriamente dito. Para os trabalhos de teste do sistema foi utilizada a versão *IAR EWB* 4.42. Ao abrir o compilador aparece uma janela inicial em que possibilita abrir um *workspace* existente (ver exemplos de *workspace)*, criar outros projetos e adicionar um projeto a um *workspace*. Clicar em *Cansel*.

Feito isso, é apresentada a interface de trabalho do programa. Deve-se ir até à guia *Project >> Create-new-project.* Selecionar a opção *Empty-project*, e clicar *OK.* Veja Figura 70.

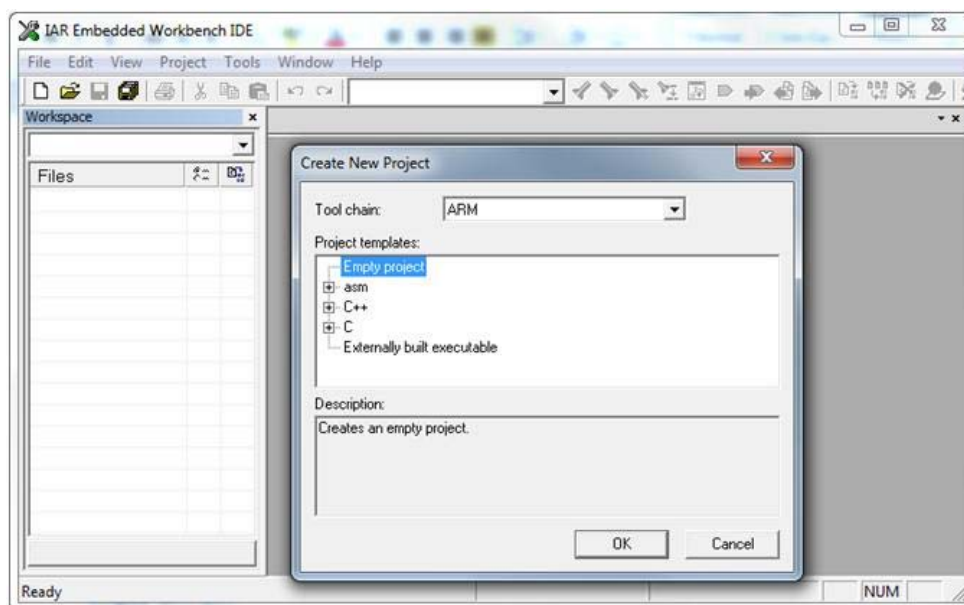



Figura 70 – Configuração inicial – *C IAR Embedded Workbench IDE*.

Na sequencia descrita anteriormente é necessário informar o *nome do projeto*. Localizar a pasta em que estão os arquivos gerados pelo *IAR MakeApp*, digitar o nome do projeto e clicar em OK. De volta à área de trabalho do programa clicar em *Project >>Add-files.* Antes de adicionar os arquivos, deve-se permitir a escolha de todos os tipos de arquivos, para isso, na seleção de *tipo,* escolha *All files (*.*).* A seguir, selecionar todos os arquivos da pasta em que o *IAR-MakeApp* gerou, e posteriormente clicar em *abrir*. Observar que o arquivo *KickStartCard_cstartup.s79* também deve ser incluído, pois sem ele o compilador não consegue compilar o programa. Acompanhe na Figura 71

[34] Web site: www.editoraerica.com.br
[35] Email : sccjunior@gmail.com

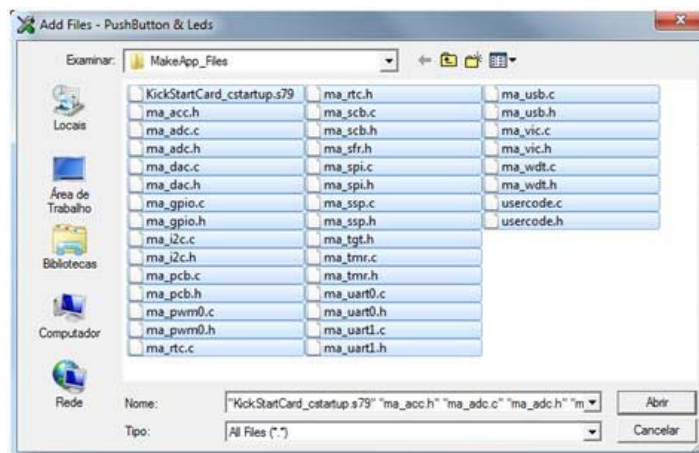



Figura 71 – Arquivos a serem adicionados – *C IAR Embedded Workbench IDE***.**

Após clicar em *Abrir*, todos os arquivos serão adicionados ao projeto e poderão ser visualizados e acessados na sessão *workspace* (no lado esquerdo da área de trabalho).

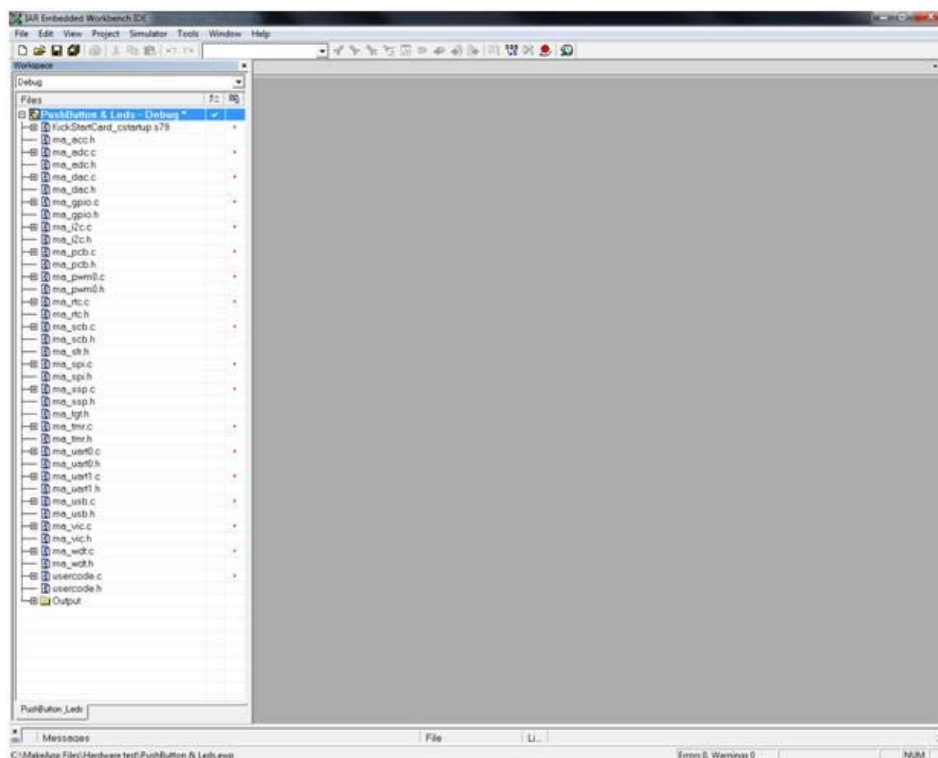



Figura 72 – Arquivos adicionados e seleção do arquivo raiz – *C IAR Embedded Workbench IDE***.**

O seguinte passo é configurar o projeto. Para tanto, selecionar o arquivo raiz do projeto (no caso apresentado, *PushButton & LEDs – Debug*, veja na Figura 72) e clicar em *Project >> Options.* Abre-se então uma janela de configuração onde vários ajustes precisam ser feitos.

No campo *Category* clicar em *General-options* e na aba *Target.* No campo *Processor-Variant,* clicar em *device.* No seletor *dropdown* do campo *Device* selecionar o componente *NXP LPC2146*. No campo *processor-mode* selecionar *ARM*. O modo *thumb* serve para otimizar a memória de programa, porém nos testes realizados não conseguiu-se implementar essa funcionalidade, devido a constantes erros (antes de finalizar a compilação do programa). Acompanhe na Figura 73.

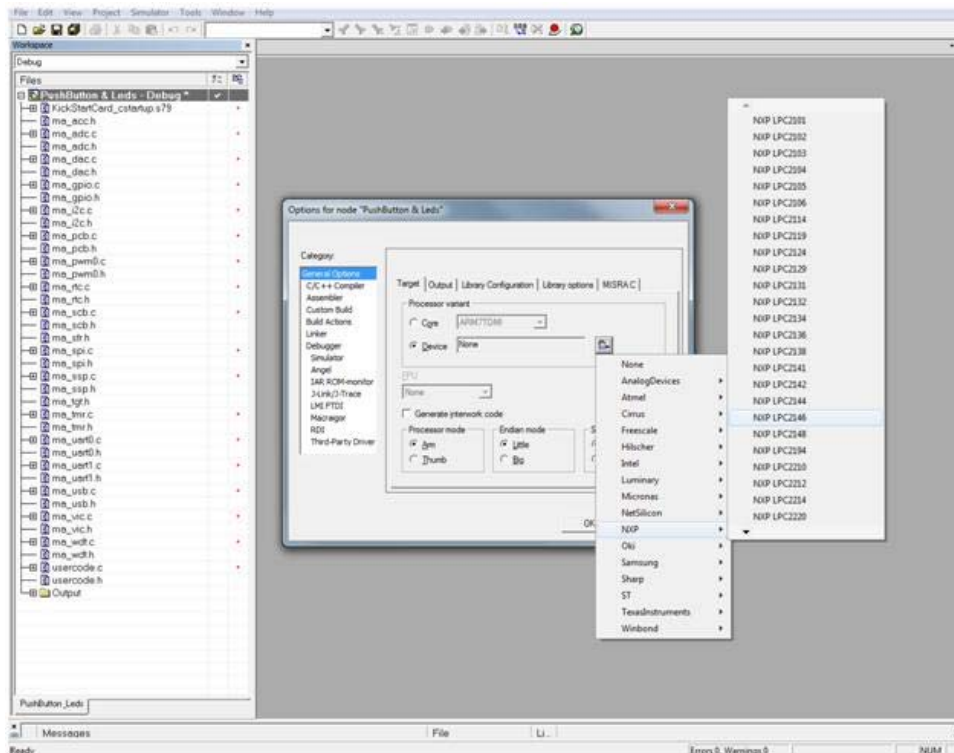



Figura 73 – Configuração do projeto: *General-options – C IAR Embedded Workbench IDE*.

Selecionar *C/C++Compiler* no campo *Category*; em seguida, na aba *Optimizations,* clicar em *Size* e selecionar a opção *None(Best-debug-suport).* Esta configuração é usada para evitar problemas ao desenvolver o programa.

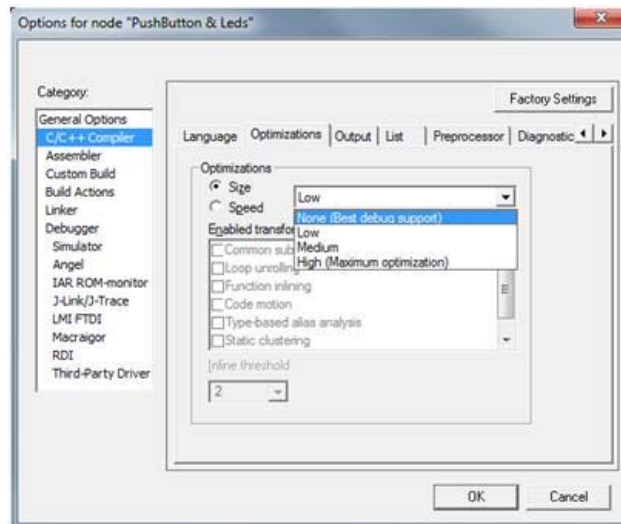



Figura 74 – Configuração do projeto: *C/C++ Compiler > Optimizations – C IAR Embedded Workbench IDE.*

Clicar na aba *List*, e selecionar *Output-list-file, e em seguida Assembler-mnemonics.*

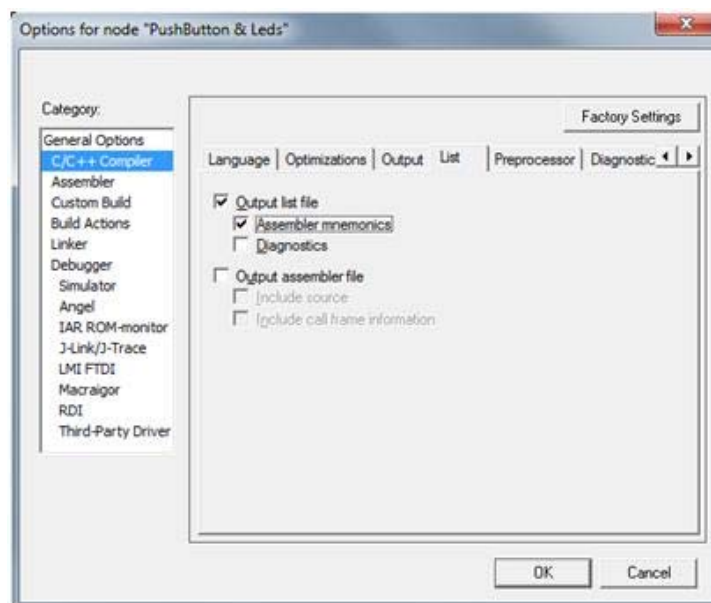



Figura 75 – Configuração do projeto: *C/C++ Compiler > List – C IAR Embedded Workbench IDE.*

Selecionar *Linker n*o campo *Category* e, na aba *Output* >> *Format,* selecionar *Allow-C-SPY*(*specific extra output file*)*.* Este item habilita as opções da aba *Extra-output* necessárias para gerar os arquivos "*.hex*" (compatíveis com microcontroladores).

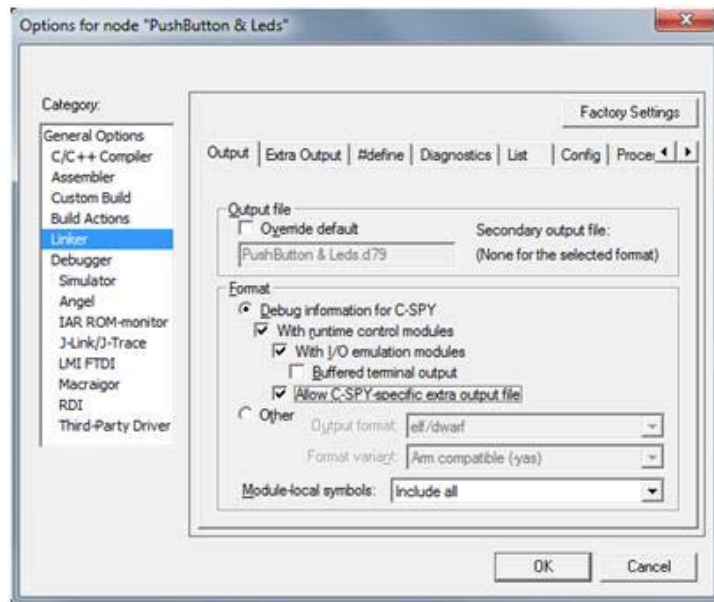



Figura 76 – Configuração do projeto: *Linker > Output – C IAR Embedded Workbench IDE.*

Posteriormente, ir até a aba *Extra-output* e selecionar *general-extra-output-files.* No campo *format* selecionar *Intel-extended* para a categoria *Output-format. Em seguida selecionar Override-default* (no campo *Output-file*) e renomear a extensão do arquivo de *".a79" para ".hex".*

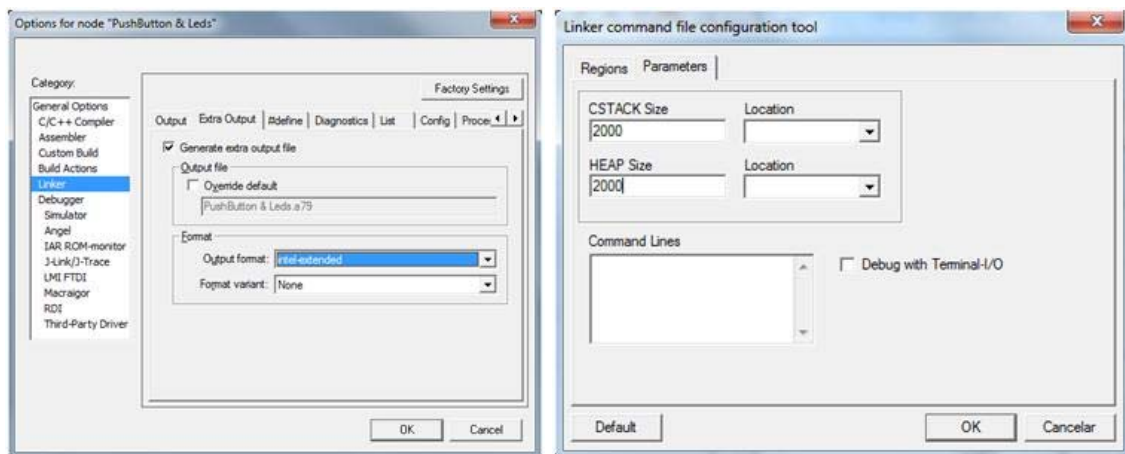



Figura 77 – Configuração do projeto: *Linker > Extra Output – C IAR Embedded Workbench IDE.*

Na aba *List*, selecionar *Generate-linker-listing.*

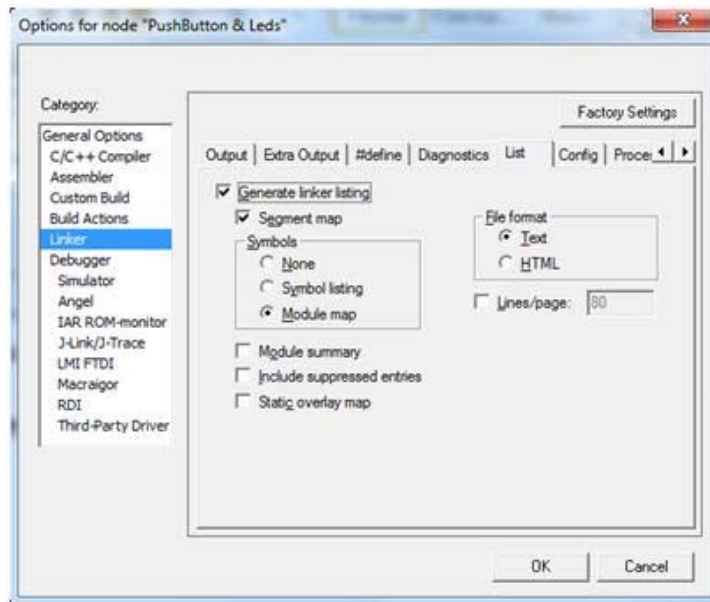



Figura 78 – Configuração do projeto: *Linker > List – C IAR Embedded Workbench IDE*.

Selecionar a aba *Config*. No campo *Linker-command-file* clicar em *Override-default,* assim como em *Command-file-configuration-tool.* Abrir-se-á uma nova janela.

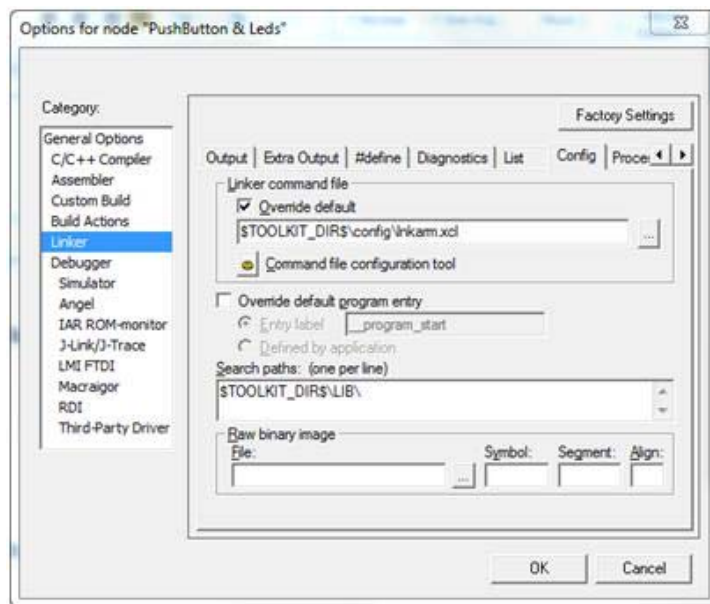



Figura 79 – Configuração do projeto: *Linker > Config – C IAR Embedded Workbench IDE*.

Na janela aberta, clicar duas vezes na inscrição *Internal-ROM-8000-FFFFF* e altere para *40-3FFF* (limitação de 32Kbytes de memória de programa). Na inscrição *internal-RAM-100000-7FFFFF* alterar o valor para *40004000-40007FFF* (uso total da memória *RAM*). Ainda nessa mesma janela, clicar na

aba *Parameters* e alterar os valores de *CSTACK-Size* e *HeapSize* para 2000. Clicar em OK.

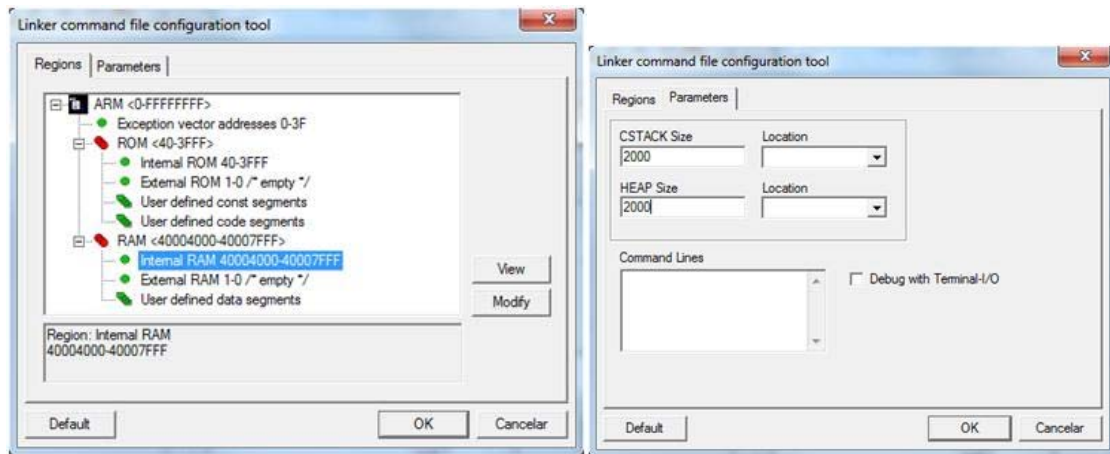



Figura 80 – Configuração do projeto: *Linker* > *List* > Linker command file configuration tool – *C IAR Embedded Workbench IDE.*

Abrir-se-á uma nova janela que deverá conter o nome e o local do arquivo de configuração que irá ser gerado. Clicar em *salvar*.

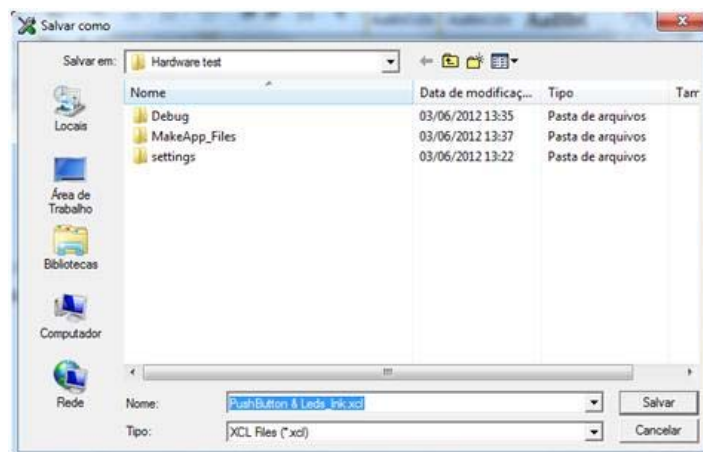



Figura 81 – Salvando arquivo de configuração – C IAR Embedded Workbench IDE**.**

Ao voltar à janela *Options,* clicar *OK. A*pós esse longo processo de configuração, o compilador está pronto para a programação. Como serão usadas as funções criadas pelo *IAR-MakeApp*, é necessário também fazer algumas alterações no ambiente do IAR antes de desenvolver o programa. No *workspace,* clicar duas vezes em *usercode.c* e após o comentário *Inicialise-*

used-periferals retirar os comentários das funções de configuração, de acordo com os periféricos que serão utilizados.

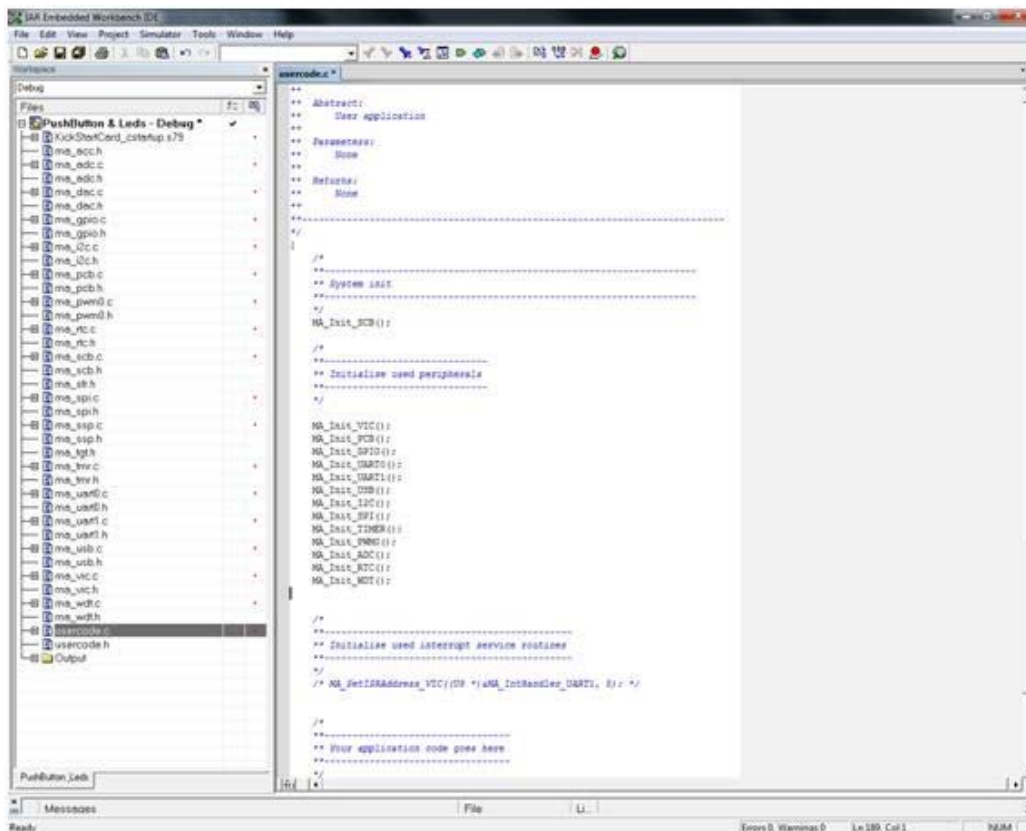



Figura 82 – Ajustes finais nas linhas de código para programação – *C IAR Embedded Workbench IDE.*

Ao finalizar o programa, basta clicar no botão *Make, ou então vá até Project >> make,* ou ainda, pressione *F7.* Caso não tenha havido algum erro após essa etapa o programa já estará compilado e o arquivo ".hex" disponibilizado na pasta do projeto. A próxima etapa é enviar esse arquivo .hex para o microprocessador utilizando a ferramenta *Flash-Magic*.

E.1.3 UTILIZANDO O FLASH-MAGIC

O processo de gravação é bem simples. Ao abrir o programa Flash-Magic, configurar a porta em que está ligada a plataforma (se necessário conferir em: \\painel de controle\sistema e segurança\sistema\gerenciador de dispositivos). Fazer o correto ajuste no campo *COM Port* ajustando o *baud rate* em 9600. No seletor *Device*, escolher *LPC2146* e no seletor *Interface*, escolher *None(ISP),* veja a Figura 83. Ajustar a frequência do oscilador em 14.7456MHz (utilizar ponto e não vírgula). Ainda nessa tela inicial, selecionar *Erase-all-flash+code-Rd-Prot.*

Conforme a necessidade do projetista, a opção *Verif-after-programming* pode ser selecionada para que seja feita uma verificação da correta programação após o processo de gravação.

Feita a configuração inicial, basta indicar o local que se encontra o arquivo *".hex"* que se pretende usar na plataforma. Normalmente ele estará onde foi indicado para serem geradas as funções no *IAR EWB.* Para transferir o programa, clicar em *start.* A Figura 83 resume todo o procedimento descrito.

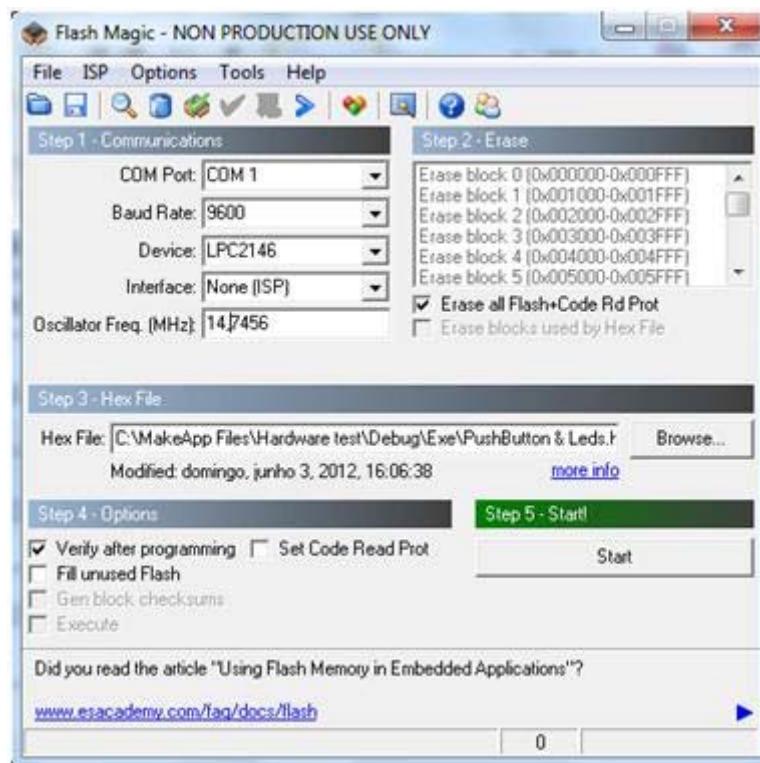



Figura 83 – Configuração *Flash Magic***.**

Vale destacar que o *jumper* no conector P12 deve estar posicionado durante o processo de programação conforme descrito na sessão E.2 a seguir.

E.2 CONFIGURAÇÃO DA PLATAFORMA PARA PROGRAMAÇÃO DO ARM7

Para se realizar a programação do *ARM7* é necessário habilitar a fonte de alimentação e o modo de programação. A fonte é habilitada quando a plataforma é ligada à rede elétrica, sendo que são utilizados *jumpers* em JP1 e JP2, conforme descrito na sessão B.1.

O modo de programação é habilitado ao inserir o *jumper* no conector P12. O *jumper* deve estar inserido ao se ligar a plataforma à rede elétrica, ou então à

cada novo ciclo de programação deve-se inserir o *jumper* e pressionar o botão S1 (próximo ao conector *Ethernet, ver sessão B.*1), o qual irá reinicializar (*reset)* o *ARM* e prepará-lo para a programação. O *jumper* no conector P12 habilita o *boot loader* para a programação *ISP*. Ligar ainda, por meio de *jumper,* os pinos 2 e 3 do conector P3 para finalizar a habilitação do *boot-loader*. Esse *jumper* em P3 pode ser deixado permanentemente durante os ciclos de programação e execução de rotina.

Para a execução do programa, deve-se retirar o *jumper* de P12 e pressionar o botão S1 (*reset*).

E.3 PROCEDIMENTO PARA PROGRAMAÇÃO DO FPGA

A programação do *FPGA* é realizada por meio da interface *JTAG*. Esta interface já vem implementada no próprio dispositivo utilizado, exigindo 6 vias de comunicação. São elas: VREF, GND, TCK, TODO, TDI e TMS.

O fabricante do *FPGA* utilizado (*Xilinx*) possui um conjunto de ferramentas para programação de seus dispositivos. Elas podem ser encontradas no próprio site do fabricante[36]. Utilizou-se o pacote *Xilinx ISE WebPACK 10.1* na fase de testes de programação. O GRACO dispõe de um *driver* de programação do mesmo fabricante, capaz de fazer a interface entre *JTAG* e *USB* (*Xilinx Platform Cable USB)* (Xilinx, 2008).

A primeira ferramenta a ser utilizada (dentro do pacote *ISE 10.1*) é o *software iMPACT.* Ele criará os arquivos de configuração e identificação dos dispositivos que serão utilizados. Ao se abrir o programa, clicar em *File* >> *New* >> *iMPACT -project.* Na janela aberta, selecionar *Create-a-new-project (.ipf)* e clicar *OK.* É então aberta uma nova janela, nela, selecionar *Configure-devices-using-Boundary-Scan* (*JTAG*), assim como *a* sub-opção *Automatically-connect-to-a-cable-and-identify-Boundary-Scan-chain*. Pressionar *Finish* (vide Figura 84).

[36] www.xilinx.com

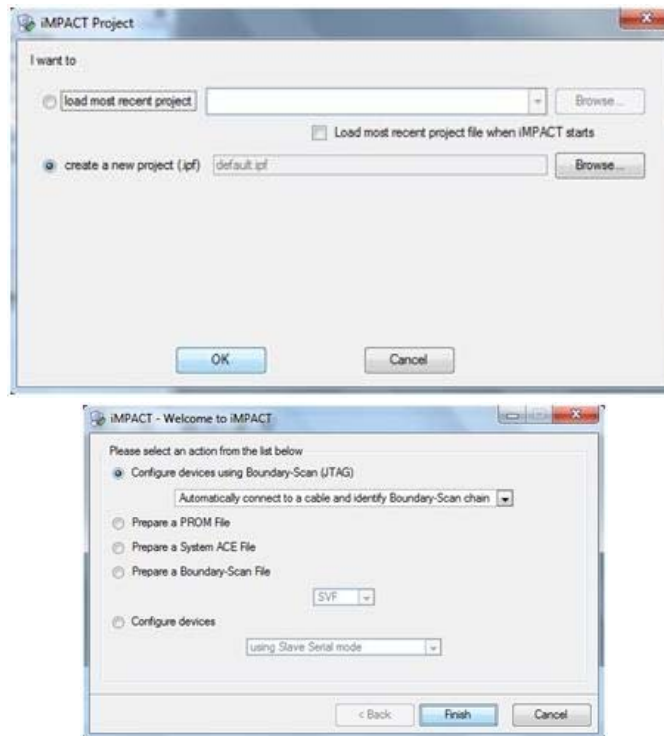



Figura 84 – Configuração *iMPACT*.

Em seguida será pedido para que seja associado um arquivo de configuração *.bit* ou *.bsd* para cada componente. Uma vez associado, pode-se fechar o *iMPACT* e a partir daí trabalhar apenas com o *ISE Project Navegator.*

E.3.1 CONFIGURAÇÃO DO XILINX ISE 10.1 PARA DESENVOLVIMENTO DE SOFTWARE

Após a correta conexão física da plataforma, basta configurar o pacote *Xilinx ISE 10.1* para começar a programar o *FPGA*. O pacote *Xilinx ISE* está disponível para *download* no próprio site do fabricante[37]. Uma vez instalado, parte-se para a configuração inicial.

A primeira coisa a se fazer no *ISE* é criar um novo projeto. Na tela inicial, clicar em *File>>Create-new-project.* Será aberta uma nova janela na qual será necessário inserir o nome do projeto e o local onde ele será criado (vide Figura 85) e clicar em *Next*.

[37] www.xilinx.com

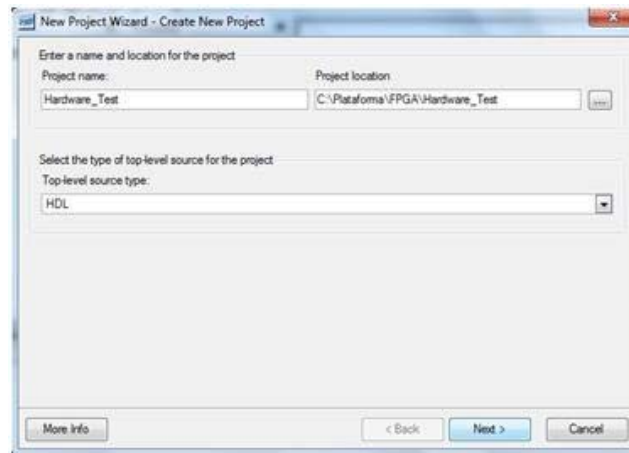



Figura 85 – Criar um novo projeto no ISE.

Em seguida, é necessário inserir a família e o dispositivo com o qual se pretende trabalhar. No caso da plataforma V1.1 é utilizado o *Spartan3E XC3S500E* no encapsulamento *PQ208*. Feita essa seleção (conforme Figura 86) clicar em *Next*.

Figura 86 – Seleção do componente no ISE.

Nas duas telas seguintes clicar em *Next* sem qualquer alteração e, ao final, clicar em *Finish* para finalizar o processo de criação de novo projeto.

Criado o projeto, é necessário adicionar uma fonte de código. Caso tenha um código pronto para ser testado, basta adicioná-lo ao projeto por meio de *Project >>Add-source.* Caso seja necessário criar uma nova fonte, basta clicar em *Project>>New-source.* Neste caso, será aberta uma nova janela, na qual deverá ser selecionado o tipo de fonte a ser criada. Nos testes realizados o tipo *VHDL-Module* foi escolhido. No lado direito é necessário inserir o nome do código a ser criado (vide Figura 87) e clicar em *Next.*

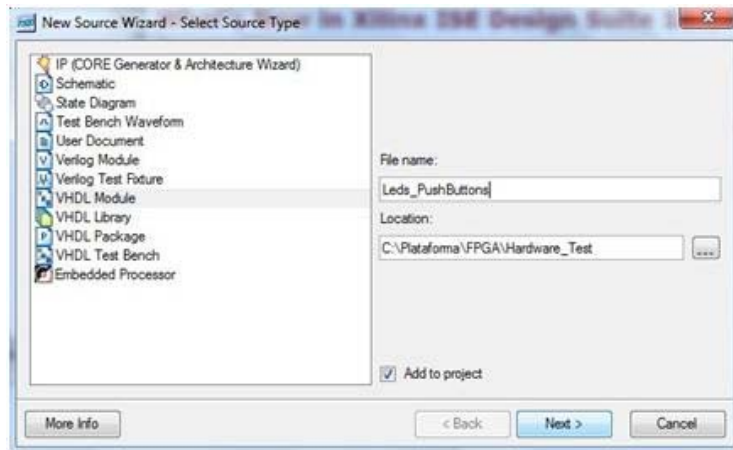



Figura 87 – Criar novo código no ISE.

Na tela seguinte, tem-se a opção de definir as entradas e saídas a serem utilizadas. Nos testes realizados, essas definições foram feitas no próprio código *VHDL*. Assim, clicar em *Next,* e na próxima janela, em *Finish.* Neste ponto, encerra-se a configuração do *Xilinx ISE* e o projetista já pode desenvolver o código livremente.

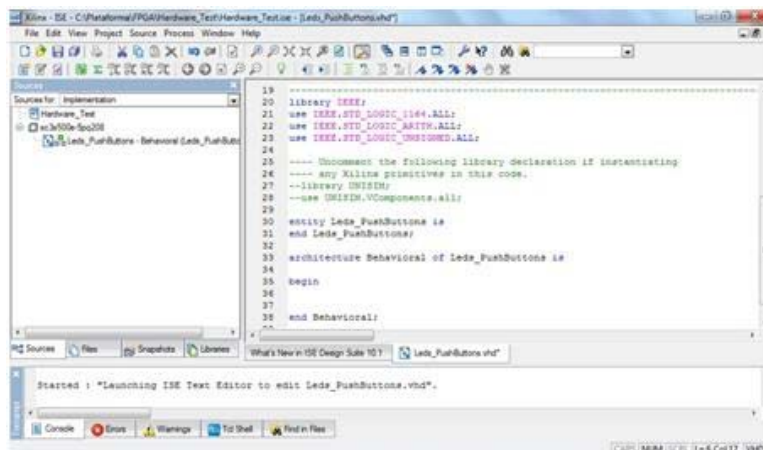



Figura 88 – Configuração final do ISE.

E.3.2 CONFIGURAÇÃO DO XILINX ISE 10.1 PARA GRAVAÇÃO

Conectar corretamente as seis vias da comunicação *JTAG* para que o pacote ISE identifique adequadamente o *FPGA* e a memória. Uma vez identificados, basta carregar o programa na plataforma por meio do pacote ISE para começar a utilizá-la. Isso pode ser feito ao clicar com o botão direito do mouse em cima do dispositivo (*FPGA* ou memória) e em seguida clicar *Program*. Se tudo estiver correto, uma mensagem aparecerá na tela: *Program-successfull*.

Diferentemente do *ARM*, o *FPGA* não necessita de ser reinicializado (*reset*) para entrar em funcionamento. Assim que finalizado o processo de gravação já se inicia a execução do código automaticamente.

E.4 CONCLUSÃO SOBRE PROCEDIMENTOS DE PROGRAMAÇÃO

Neste anexo, foram destacados todos os procedimentos para programação da plataforma ora desenvolvida. Essa parte descritiva possui grande relevância aqueles que pretenderem dar continuidade ao projeto ou mesmo utilizar a plataforma para desenvolver soluções de engenharia.

Nota-se que são muitas configurações e em diferentes plataformas. Uma boa documentação do processo de programação torna-se essencial especialmente para aqueles que estiverem iniciando o desenvolvimento, seja do *ARM* seja do *FPGA*.

Sem dúvida, esse relato detalhado e minucioso servirá de guia para os utilizadores da plataforma. Com isso, possibilita a fácil utilização e melhorias de projeto.

Além disso, esse nível de documentação contribui para a solidificação e transferência do conhecimento adquirido durante as fases de desenvolvimento para que a equipe do GRACO tenha totais condições de operação e modificação da plataforma.

APÊNDICE F. CÓDIGOS UTILIZADOS PARA OS TESTES DA PLATAFORMA

F.1 CÓDIGO C UTILIZADO PARA O TESTE DE FUNCIONAMENTO DO ARM7 (ISOLADO)

```
/*
*****************************************************************************
**
**    Project     : MsC
**
**    Component   : MakeUpARM (LPC2148)
**
**    Modulename  : System
**
**    Filename    : template.c
**
**    Abstract    : This file is the template file for an application
**                  Rename the file from <template.c> to e.g. <usercode.c>
**                  The file contains an example of main function
**
**                  Modify the <usercode.c> to suit your application.
**                  The <usercode.c> file will not be overwritten by MakeApp.
**
**    Date        : 2012-05-21 20:28:49
**
**    License no. : 9530-960-683-7130     Samuel Jr
**
**    Warning     : This file has been automatically generated.
**                  Do not edit this file if you intend to regenerate it.
**
**    This template file was created by IAR MakeApp version
**    4.10C (Philips LPC2138-48: 4.01B) for the Philips LPC2148 series of
**    microcontrollers.
**
**    (c)Copyright 2004-2005 MakeApp Consulting.
**
*****************************************************************************
*/
/*
**=========================================================================
====
** 1.2     References
**
** No   Identification          Name or Description
** ==   ===================     ================================
** 1    Rev 0.1 - 22 June 2005  Philips LPC2142/2148
```

```
**                        Preliminary data sheet
**  2    2004 Sep 24            Philips ARM-LPC201xU Objective spec
**                        Philips USB device controller (IP_3503)
**
**=========================================================================
====
*/

#include "usercode.h"   /* Usercode macros (see <template.h>) */
#include "ma_tgt.h"     /* Target specific header file */
#include "ma_sfr.h"     /* Special function register bitfield macros */
#include "iolpc2148.h"  /* Defines Special function registers */

#include "ma_scb.h"
#include "ma_gpio.h"

/*
**=========================================================================
====
**  3.      DECLARATIONS
**  3.1     Internal constants
**=========================================================================
====
*/
#define   D6 0x00010000
#define   D7 0x00080000
#define   D8 0x00040000
#define   D9 0x00020000
#define   D10 0x00100000


/*
**=========================================================================
====
**  3.2     Internal macros
**=========================================================================
====
*/
#define   LIGA_LED1(void)     MA_WritePort_GPIO(1,0xFFFFFFFF,D7)
#define   DESLIGA_LED1(void)  MA_WritePort_GPIO(1,0x00000000,D7)

#define   LIGA_LED2(void)     MA_WritePort_GPIO(1,0xFFFFFFFF,D8)
#define   DESLIGA_LED2(void)  MA_WritePort_GPIO(1,0x00000000,D8)

#define   LIGA_LED3(void)     MA_WritePort_GPIO(1,0xFFFFFFFF,D9)
#define   DESLIGA_LED3(void)  MA_WritePort_GPIO(1,0x00000000,D9)

#define   LIGA_LED4(void)     MA_WritePort_GPIO(1,0xFFFFFFFF,D6)
#define   DESLIGA_LED4(void)  MA_WritePort_GPIO(1,0x00000000,D6)

#define   LIGA_LED5(void)     MA_WritePort_GPIO(1,0xFFFFFFFF,D10)
```

```
#define   DESLIGA_LED5(void)  MA_WritePort_GPIO(1,0x00000000,D10)

#define   BT2
#define   BT3
#define   BT4
#define   BT5
#define   BT6

/*
**=====================================================================
====
**  3.3     Internal type definitions
**=====================================================================
====
*/
union port
{
  struct
    {
        unsigned int :21;
        unsigned int BIT21:1;
        unsigned int BIT22:1;
        unsigned int BIT23:1;
        unsigned int BIT24:1; // Lixo
        unsigned int BIT25:1;
        unsigned int :6;
    } ;

    struct
    {
        unsigned int :21;
        unsigned int BOTOES:5;
        unsigned int :6;
    } ;

    struct
    {
        unsigned int BITS;
    } ;

};

union port PORT_COPY;
#define BOTAO_1   PORT_COPY.BIT21
#define BOTAO_2   PORT_COPY.BIT22
#define BOTAO_3   PORT_COPY.BIT23   //O 24 é lixo.
#define BOTAO_4   PORT_COPY.BIT25
#define BOTAO_5   PORT_COPY.BIT30

#define BOT_ALL   PORT_COPY.BOTOES
```

```
void main( void )
/*
**-----------------------------------------------------------------------------
**
**  Abstract:
**      User application
**
**  Parameters:
**      None
**
**  Returns:
**      None
**
**-----------------------------------------------------------------------------
*/
{
    /*
    **------------------------------------------------------------------
    ** System init
    **------------------------------------------------------------------
    */
    MA_Init_SCB();

    /*
    **-----------------------------
    ** Initialise used peripherals
    **-----------------------------
    */
    MA_Init_GPIO();

    /*
    **---------------------------------
    ** Your application code goes here
    **---------------------------------
    */
    while( 1 )
    {
         PORT_COPY.BITS = MA_ReadPort_GPIO(1);
        //PORT_COPY.BITS = MA_ReadPort_GPIO(0);


       if(!BOTAO_1) LIGA_LED1();
        else DESLIGA_LED1();

       if(!BOTAO_2) LIGA_LED2();
        else DESLIGA_LED2();


       if(!BOTAO_3) LIGA_LED3();
        else DESLIGA_LED3();
```

```
      if(!BOTAO_4) LIGA_LED4();
       else DESLIGA_LED4();
    }

} /* main */
```

F.2 CÓDIGO VHDL UTILIZADO PARA O TESTE DE FUNCIONAMENTO DO FPGA (ISOLADO)

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    14:07:00 05/16/2012
-- Design Name:
-- Module Name:    teste_not - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity teste_not is
port(
                BT12 : in std_logic;
                BT13 : in std_logic;
                BT7  : in std_logic;
                BT8  : in std_logic;
                BT9  : in std_logic;
                BT10 : in std_logic;
```

```
                    BT11 : in std_logic;
                    BT14 : in std_logic;
                    Botao: in std_logic_vector (7 downto 0);

                    D23 : out std_logic;
                    D22 : out std_logic);
                    D21 : out std_logic;
                    D20 : out std_logic;
                    D19 : out std_logic;
                    D24 : out std_logic;
                    D25 : out std_logic;
                    D26 : out std_logic);
                    LED: out std_logic_vector (7 downto 0));


end teste_not;

architecture Behavioral of teste_not is

begin

          D23 <= not BT12;
          D22 <= not BT13;
          D21 <= not BT7;
          D20 <= not BT8;
          D19 <= not BT9;
          D24 <= not BT10;
          D25 <= not BT11;
          D26 <= not BT14;

end Behavioral;
```

Arquivo teste_not.ucf:

```
#PACE: Start of Constraints generated by PACE
#PACE: Start of PACE I/O Pin Assignments

#NET "BT7"  LOC = "p77";
#NET "BT8"  LOC = "p83";
NET "BT9"  LOC = "p84";
#NET "BT10"  LOC = "p86";
#NET "BT11"  LOC = "p87";
#NET "BT12"  LOC = "p69";
#NET "BT13"  LOC = "p74";
#NET "BT12"  LOC = "p98";

NET "D19"  LOC = "p97";
NET "D20"  LOC = "p96";
```

```
NET "D21"  LOC = "p94";
NET "D22"  LOC = "p93";
NET "D23"  LOC = "p89";
NET "D24"  LOC = "p99";
NET "D25"  LOC = "p100";
NET "D26"  LOC = "p102";

#PACE: Start of PACE Area Constraints
#PACE: Start of PACE Prohibit Constraints
#PACE: End of Constraints generated by PACE
```

F.3 CÓDIGO C UTILIZADO PARA O TESTE DE FUNCIONAMENTO DO ARM7 INTEGRADO AO FPGA

```
#include "usercode.h"   /* Usercode macros (see <template.h>) */
#include "ma_tgt.h"     /* Target specific header file */
#include "ma_sfr.h"     /* Special function register bitfield macros */
#include "iolpc2148.h"  /* Defines Special function registers */

#include "ma_scb.h"
#include "ma_gpio.h"

#define   D9  0x00020000
#define   P0_0 1

#define   LIGA_FPGA_JP4(void)     MA_WritePort_GPIO(0,0xFFFFFFFF,P0_0)
#define   DESLIGA_FPGA_JP4(void)  MA_WritePort_GPIO(0,0x00000000,P0_0)

#define   LIGA_LED3(void)     MA_WritePort_GPIO(1,0xFFFFFFFF,D9)
#define   DESLIGA_LED3(void)  MA_WritePort_GPIO(1,0x00000000,D9)

union port
{
  struct
    {
       unsigned int :21;
       unsigned int BIT21:1;
       unsigned int BIT22:1;
       unsigned int BIT23:1;
       unsigned int BIT24:1; // Lixo
       unsigned int BIT25:1;
       unsigned int :6;
    } ;

    struct
    {
       unsigned int BITS_1;
    } ;
```

```
};

union port PORT_COPY;
#define BOTAO_1   PORT_COPY.BIT21

union port0
{
  struct
    {
        unsigned int :1;
        unsigned int BIT1:1;
        unsigned int :28;// Lixo
        unsigned int BIT30:1;
        unsigned int :1;
     } ;

    struct
    {
        unsigned int BITS_0;
    } ;
};

union port0 PORT_COPY_0;
#define BOTAO_0   PORT_COPY_0.BIT1
#define BOTAO_6   PORT_COPY_0.BIT30

void main( void )
{
    MA_Init_SCB();
    MA_Init_GPIO();
    while( 1 )
    {
         PORT_COPY.BITS_1   = MA_ReadPort_GPIO(1);
         PORT_COPY_0.BITS_0 = MA_ReadPort_GPIO(0);

       if(!BOTAO_1) LIGA_FPGA_JP4();
        else DESLIGA_FPGA_JP4(); // Se apertar o btn1, liga FPGA

       if(!BOTAO_0) LIGA_LED3();
        else DESLIGA_LED3();  // Se receber FPGA, liga led 1

    }
} /* main */
```

F.4 CÓDIGO VHDL UTILIZADO PARA O TESTE DE FUNCIONAMENTO DO FPGA INTEGRADO AO ARM

```
--------------------------------------------------------------------------------
```

```
-- Company:
-- Engineer:
--
-- Create Date:    14:07:00 05/16/2012
-- Design Name:
-- Module Name:    teste_not - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity teste_not is
port(
	BT7,BT8,BT9,BT10,BT11,BT12,BT13,BT14,FROM_ARM  : in std_logic;
	D19,D20,D21,D22,D23,D24,D25,D26, TO_ARM       : out std_logic);

end teste_not;

architecture Behavioral of teste_not is

begin

D22<= FROM_ARM;
TO_ARM <= BT9;

D23<= '0';
D21<= '0';
D20<= '0';
D19<= '0';
D24<= '0';
D25<= '0';
D26<= '0';

end Behavioral;
```

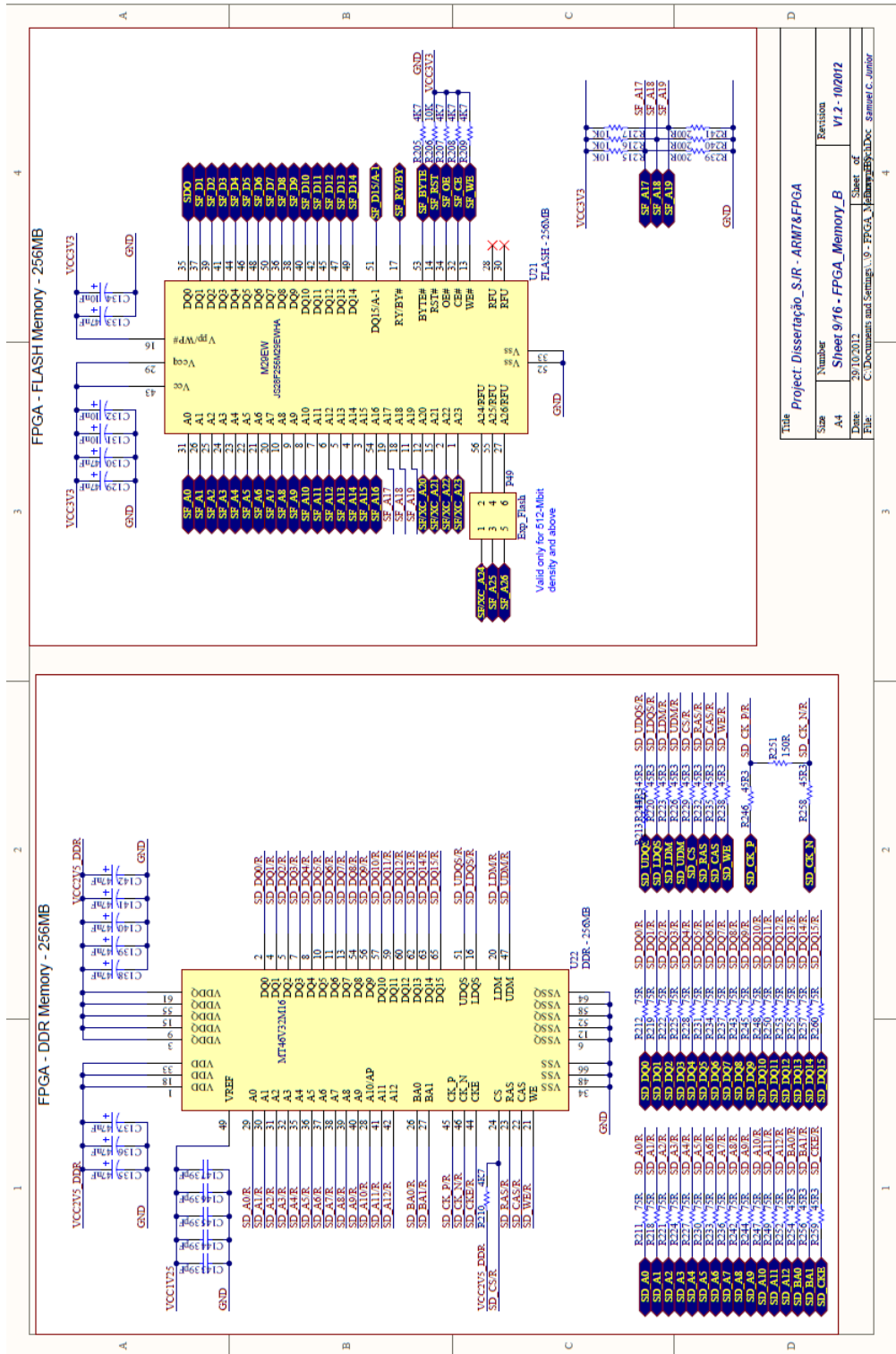
Arquivo teste_not.ucf:

```
#PACE: Start of Constraints generated by PACE

#PACE: Start of PACE I/O Pin Assignments

#NET "BT7"  LOC = "p77";
#NET "BT8"  LOC = "p83";
NET "BT9"  LOC = "p84";
#NET "BT10"  LOC = "p86";
#NET "BT11"  LOC = "p87";
#NET "BT12"  LOC = "p69";
#NET "BT13"  LOC = "p74";
#NET "BT12"  LOC = "p98";

NET "D19"  LOC = "p97";
NET "D20"  LOC = "p96";
NET "D21"  LOC = "p94";
NET "D22"  LOC = "p93";
NET "D23"  LOC = "p89";
NET "D24"  LOC = "p99";
NET "D25"  LOC = "p100";
NET "D26"  LOC = "p102";

NET "FROM_ARM"  LOC = "p171";
NET "TO_ARM"    LOC = "p168";

#PACE: Start of PACE Area Constraints

#PACE: Start of PACE Prohibit Constraints

#PACE: End of Constraints generated by PACE
```
[i]

APÊNDICE G. CIRCUITOS ELETRÔNICOS (VERSÃO 1.2)

FPGA - Power Input
J9
Jack - PWR
Up to 9V
SW2
SW-SPST
D23
Bridge
D24
VCC9V
U23
VR_5V
Vin
Vout
GND
D22
1N4148
C226
470uF
C227
0.1uF
C228
C229
D20
1N5819
VCC5VA
D21
USB_5V
JP18
EN_5V
5V - 500mA
VCC5V
D25
D Zener 5V
GND
Title
Project: Dissertação_SJR - ARM7&FPGA
Size
A4
Number
Sheet 6/16 - FPGA_PWR Input
Revision
V1.2 - 10/2012
Date:
29/10/2012
Sheet of
File:
Samuel C. Junior

FPGA - Serial Interface
Flash PROM 4Mb
EEPROM 20KBIT
Title
Project: Dissertação_SJR - ARM7&FPGA
Size
A4
Number
Sheet 8/16 - FPGA_Memory_A
Revision
V1.2 - 10/2012
Date: 29/10/2012
Samuel C. Junior

FPGA - FLASH Memory - 256MB
FPGA - DDR Memory - 256MB
Project: Dissertação_SJR - ARM7&FPGA
Sheet 9/16 - FPGA_Memory_B
V1.2 - 10/2012
Samuel C. Junior

FPGA - Cool Runner
Project: Dissertação_SJR - ARM7&FPGA
Sheet 10/16 - FPGA_Cool Runner
V1.2 - 10/2012
A4
29/10/2012

FPGA - Communication - RS232
FPGA - VGA Port
FPGA - USB
FPGA - PS2 Port
MAX3232
Project: Dissertação_SJR - ARM7&FPGA
Sheet 12/16 - FPGA_Connections
V1.2 - 10/2012
29/10/2012
Samuel C. Junior

Project: Dissertação_SJR - ARM7&FPGA
Sheet 13/16 - FPGA_Ethernet
V1.2 - 10/2012
A4
29/10/2012
Samuel C. Junior
LAN83C185
U18
RJ45
J5
25MHz